* 1. **Question 1. What Is Spark?**

**Answer :**

Most of the data users know only SQL and are not good at programming. Spark is a tool, developed for people who are from a database background - to access Scala MLib capabilities through Hive like SQL interface. Spark tool helps data users run Hive on Spark - offering compatibility with Hive metastore, queries and data.

* 1. **Question 2. Most Of The Data Users Know Only Sql And Are Not Good At Programming. Spark Is A Tool, Developed For People Who Are From A Database Background - To Access Scala Mlib Capabilities Through Hive Like Sql Interface. Spark Tool Helps Data Users Run Hive On Spark - Offering Compatibility With Hive Metastore, Queries And Data.**

**Answer :**

* + 1. Sensor Data Processing –Apache Spark’s ‘In-memory computing’ works best here, as data is retrieved and combined from different sources.
    2. Spark is preferred over Hadoop for real time querying of data
    3. Stream Processing – For processing logs and detecting frauds in live streams for alerts, Apache Spark is the best solution.

[Python Interview Questions](https://www.wisdomjobs.com/e-university/python-interview-questions.html)

* 1. **Question 3. What Is A Sparse Vector?**

**Answer :**

sparse vector has two parallel arrays –one for indices and the other for values. These vectors are used for storing non-zero entries to save space.

* 1. **Question 4. What Is Rdd?**

**Answer :**

RDDs (Resilient Distributed Datasets) are basic abstraction in Apache Spark that represent the data coming into the system in object format. RDDs are used for in-memory computations on large clusters, in a fault tolerant manner. RDDs are read-only portioned, collection of records, that are –

* + 1. Immutable – RDDs cannot be altered.
    2. Resilient – If a node holding the partition fails the other node takes the data.
  1. **Question 5. Explain About Transformations And Actions In The Context Of Rdds.**

**Answer :**

Transformations are functions executed on demand, to produce a new RDD. All transformations are followed by actions. Some examples of transformations include map, filter and reduceByKey.

Actions are the results of RDD computations or transformations. After an action is performed, the data from RDD moves back to the local machine. Some examples of actions include reduce, collect, first, and take.

* 1. **Question 6. What Are The Languages Supported By Apache Spark For Developing Big Data Applications?**

**Answer :**

Scala, Java, Python, R and Clojure

* 1. **Question 7. Can You Use Spark To Access And Analyse Data Stored In Cassandra Databases?**

**Answer :**

Yes, it is possible if you use Spark Cassandra Connector.

* 1. **Question 8. Is It Possible To Run Apache Spark On Apache Mesos?**

**Answer :**

Yes, Apache Spark can be run on the hardware clusters managed by Mesos.

* 1. **Question 9. Explain About The Different Cluster Managers In Apache Spark**

**Answer :**

The 3 different clusters managers supported in Apache Spark are:

* + 1. YARN
    2. Apache Mesos -Has rich resource scheduling capabilities and is well suited to run Spark along with other applications. It is advantageous when several users run interactive shells because it scales down the CPU allocation between commands.
    3. Standalone deployments – Well suited for new deployments which only run and are easy to set up.
  1. **Question 10. How Can Spark Be Connected To Apache Mesos?**

**Answer :**

To connect Spark with Mesos-

* + 1. Configure the spark driver program to connect to Mesos. Spark binary package should be in a location accessible by Mesos. (or)
    2. Install Apache Spark in the same location as that of Apache Mesos and configure the property ‘spark.mesos.executor.home’ to point to the location where it is installed.
  1. **Question 11. How Can You Minimize Data Transfers When Working With Spark?**

**Answer :**

Minimizing data transfers and avoiding shuffling helps write spark programs that run in a fast and reliable manner. The various ways in which data transfers can be minimized when working with Apache Spark are:

* + 1. Using Broadcast Variable- Broadcast variable enhances the efficiency of joins between small and large RDDs.
    2. Using Accumulators – Accumulators help update the values of variables in parallel while executing.
    3. The most common way is to avoid operations ByKey, repartition or any other operations which trigger shuffles.
  1. **Question 12. Why Is There A Need For Broadcast Variables When Working With Apache Spark?**

**Answer :**

These are read only variables, present in-memory cache on every machine. When working with Spark, usage of broadcast variables eliminates the necessity to ship copies of a variable for every task, so data can be processed faster. Broadcast variables help in storing a lookup table inside the memory which enhances the retrieval efficiency when compared to an RDD lookup ().

* 1. **Question 13. Is It Possible To Run Spark And Mesos Along With Hadoop?**

**Answer :**

Yes, it is possible to run Spark and Mesos with Hadoop by launching each of these as a separate service on the machines. Mesos acts as a unified scheduler that assigns tasks to either Spark or Hadoop.

* 1. **Question 14. What Is Lineage Graph?**

**Answer :**

The RDDs in Spark, depend on one or more other RDDs. The representation of dependencies in between RDDs is known as the lineage graph. Lineage graph information is used to compute each RDD on demand, so that whenever a part of persistent RDD is lost, the data that is lost can be recovered using the lineage graph information.

* 1. **Question 15. How Can You Trigger Automatic Clean-ups In Spark To Handle Accumulated Metadata?**

**Answer :**

You can trigger the clean-ups by setting the parameter ‘spark.cleaner.ttl’ or by dividing the long running jobs into different batches and writing the intermediary results to the disk.

* 1. **Question 16. Explain About The Major Libraries That Constitute The Spark Ecosystem**

**Answer :**

* + 1. Spark MLib- Machine learning library in Spark for commonly used learning algorithms like clustering, regression, classification, etc.
    2. Spark Streaming – This library is used to process real time streaming data.
    3. Spark GraphX – Spark API for graph parallel computations with basic operators like joinVertices, subgraph, aggregateMessages, etc.
    4. Spark SQL – Helps execute SQL like queries on Spark data using standard visualization or BI tools.
  1. **Question 17. What Are The Benefits Of Using Spark With Apache Mesos?**

**Answer :**

It renders scalable partitioning among various Spark instances and dynamic partitioning between Spark and other big data frameworks.

* 1. **Question 18. What Is The Significance Of Sliding Window Operation?**

**Answer :**

Sliding Window controls transmission of data packets between various computer networks. Spark Streaming library provides windowed computations where the transformations on RDDs are applied over a sliding window of data. Whenever the window slides, the RDDs that fall within the particular window are combined and operated upon to produce new RDDs of the windowed DStream.

* 1. **Question 19. What Is A Dstream?**

**Answer :**

Discretized Stream is a sequence of Resilient Distributed Databases that represent a stream of data. DStreams can be created from various sources like Apache Kafka, HDFS, and Apache Flume. DStreams have two operations –

* + 1. Transformations that produce a new DStream.
    2. Output operations that write data to an external system.
  1. **Question 20. When Running Spark Applications, Is It Necessary To Install Spark On All The Nodes Of Yarn Cluster?**

**Answer :**

Spark need not be installed when running a job under YARN or Mesos because Spark can execute on top of YARN or Mesos clusters without affecting any change to the cluster.

* 1. **Question 21. What Is Catalyst Framework?**

**Answer :**

Catalyst framework is a new optimization framework present in Spark SQL. It allows Spark to automatically transform SQL queries by adding new optimizations to build a faster processing system.

* 1. **Question 22. Name A Few Companies That Use Apache Spark In Production.**

**Answer :**

Pinterest, Conviva, Shopify, Open Table

* 1. **Question 23. Which Spark Library Allows Reliable File Sharing At Memory Speed Across Different Cluster Frameworks?**

**Answer :**

Tachyon

Work On Interesting Data Science Projects using Spark to build an impressive project portfolio!

* 1. **Question 24. Why Is Blinkdb Used?**

**Answer :**

BlinkDB is a query engine for executing interactive SQL queries on huge volumes of data and renders query results marked with meaningful error bars. BlinkDB helps users balance ‘query accuracy’ with response time.

* 1. **Question 25. How Can You Compare Hadoop And Spark In Terms Of Ease Of Use?**

**Answer :**

Hadoop MapReduce requires programming in Java which is difficult, though Pig and Hive make it considerably easier. Learning Pig and Hive syntax takes time. Spark has interactive APIs for different languages like Java, Python or Scala and also includes Spark i.e. Spark SQL for SQL lovers - making it comparatively easier to use than Hadoop.

* 1. **Question 26. What Are The Common Mistakes Developers Make When Running Spark Applications?**

**Answer :**

Developers often make the mistake of-

* + 1. Hitting the web service several times by using multiple clusters.
    2. Run everything on the local node instead of distributing it.

Developers need to be careful with this, as Spark makes use of memory for processing.

* 1. **Question 27. What Is The Advantage Of A Parquet File?**

**Answer :**

Parquet file is a columnar format file that helps –

* + 1. Limit I/O operations
    2. Consumes less space
    3. Fetches only required columns.
  1. **Question 28. What Are The Various Data Sources Available In Sparksql?**

**Answer :**

* + 1. Parquet file
    2. JSON Datasets
    3. Hive tables
  1. **Question 29. How Spark Uses Hadoop?**

**Answer :**

Spark has its own cluster management computation and mainly uses Hadoop for storage.

* 1. **Question 30. What Are The Key Features Of Apache Spark That You Like?**

**Answer :**

* + 1. Spark provides advanced analytic options like graph algorithms, machine learning, streaming data, etc
    2. It has built-in APIs in multiple languages like Java, Scala, Python and R
    3. It has good performance gains, as it helps run an application in the Hadoop cluster ten times faster on disk and 100 times faster in memory.
  1. **Question 31. What Do You Understand By Pair Rdd?**

**Answer :**

Special operations can be performed on RDDs in Spark using key/value pairs and such RDDs are referred to as Pair RDDs. Pair RDDs allow users to access each key in parallel. They have a reduceByKey () method that collects data based on each key and a join () method that combines different RDDs together, based on the elements having the same key.

* 1. **Question 32. Which One Will You Choose For A Project –hadoop Mapreduce Or Apache Spark?**

**Answer :**

As it is known that Spark makes use of memory instead of network and disk I/O. However, Spark uses large amount of RAM and requires dedicated machine to produce effective results. So the decision to use Hadoop or Spark varies dynamically with the requirements of the project and budget of the organization.

* 1. **Question 33. Explain About The Different Types Of Transformations On Dstreams?**

**Answer :**

* + 1. Stateless Transformations- Processing of the batch does not depend on the output of the previous batch. Examples – map (), reduceByKey (), filter ().
    2. Stateful Transformations- Processing of the batch depends on the intermediary results of the previous batch. Examples –Transformations that depend on sliding windows.
  1. **Question 34. Explain About The Popular Use Cases Of Apache Spark**

**Answer :**

Apache Spark is mainly used for

* + 1. Iterative machine learning.
    2. Interactive data analytics and processing.
    3. Stream processing
    4. Sensor data processing
  1. **Question 35. Is Apache Spark A Good Fit For Reinforcement Learning?**

**Answer :**

No. Apache Spark works well only for simple machine learning algorithms like clustering, regression, classification.

* 1. **Question 36. What Is Spark Core?**

**Answer :**

It has all the basic functionalities of Spark, like - memory management, fault recovery, interacting with storage systems, scheduling tasks, etc.

* 1. **Question 37. How Can You Remove The Elements With A Key Present In Any Other Rdd?**

**Answer :**

Use the subtractByKey () function

* 1. **Question 38. What Is The Difference Between Persist() And Cache()**

**Answer :**

persist () allows the user to specify the storage level whereas cache () uses the default storage level.

* 1. **Question 39. What Are The Various Levels Of Persistence In Apache Spark?**

**Answer :**

Apache Spark automatically persists the intermediary data from various shuffle operations, however it is often suggested that users call persist () method on the RDD in case they plan to reuse it. Spark has various persistence levels to store the RDDs on disk or in memory or as a combination of both with different replication levels.

The various storage/persistence levels in Spark are -

* + 1. MEMORY\_ONLY
    2. MEMORY\_ONLY\_SER
    3. MEMORY\_AND\_DISK
    4. MEMORY\_AND\_DISK\_SER, DISK\_ONLY
    5. OFF\_HEAP
  1. **Question 40. How Spark Handles Monitoring And Logging In Standalone Mode?**

**Answer :**

Spark has a web based user interface for monitoring the cluster in standalone mode that shows the cluster and job statistics. The log output for each job is written to the work directory of the slave nodes.

* 1. **Question 41. Does Apache Spark Provide Check Pointing?**

**Answer :**

Lineage graphs are always useful to recover RDDs from a failure but this is generally time consuming if the RDDs have long lineage chains. Spark has an API for check pointing i.e. a REPLICATE flag to persist. However, the decision on which data to checkpoint - is decided by the user. Checkpoints are useful when the lineage graphs are long and have wide dependencies.

* 1. **Question 42. How Can You Launch Spark Jobs Inside Hadoop Mapreduce?**

**Answer :**

Using SIMR (Spark in MapReduce) users can run any spark job inside MapReduce without requiring any admin rights.

* 1. **Question 43. How Spark Uses Akka?**

**Answer :**

Spark uses Akka basically for scheduling. All the workers request for a task to master after registering. The master just assigns the task. Here Spark uses Akka for messaging between the workers and masters.

* 1. **Question 44. How Can You Achieve High Availability In Apache Spark?**

**Answer :**

* + 1. Implementing single node recovery with local file system
    2. Using StandBy Masters with Apache ZooKeeper.
  1. **Question 45. Hadoop Uses Replication To Achieve Fault Tolerance. How Is This Achieved In Apache Spark?**

**Answer :**

Data storage model in Apache Spark is based on RDDs. RDDs help achieve fault tolerance through lineage. RDD always has the information on how to build from other datasets. If any partition of a RDD is lost due to failure, lineage helps build only that particular lost partition.

* 1. **Question 46. Explain About The Core Components Of A Distributed Spark Application.**

**Answer :**

* + 1. Driver- The process that runs the main () method of the program to create RDDs and perform transformations and actions on them.
    2. Executor –The worker processes that run the individual tasks of a Spark job.
    3. Cluster Manager-A pluggable component in Spark, to launch Executors and Drivers. The cluster manager allows Spark to run on top of other external managers like Apache Mesos or YARN.
  1. **Question 47. What Do You Understand By Lazy Evaluation?**

**Answer :**

Spark is intellectual in the manner in which it operates on data. When you tell Spark to operate on a given dataset, it heeds the instructions and makes a note of it, so that it does not forget - but it does nothing, unless asked for the final result. When a transformation like map () is called on a RDD-the operation is not performed immediately. Transformations in Spark are not evaluated till you perform an action. This helps optimize the overall data processing workflow.

* 1. **Question 48. Define A Worker Node.**

**Answer :**

A node that can run the Spark application code in a cluster can be called as a worker node. A worker node can have more than one worker which is configured by setting the SPARK\_ WORKER\_INSTANCES property in the spark-env.sh file. Only one worker is started if the SPARK\_ WORKER\_INSTANCES property is not defined.

* 1. **Question 49. What Do You Understand By Schemardd?**

**Answer :**

An RDD that consists of row objects (wrappers around basic string or integer arrays) with schema information about the type of data in each column.

* 1. **Question 50. What Are The Disadvantages Of Using Apache Spark Over Hadoop Mapreduce?**

**Answer :**

Apache spark does not scale well for compute intensive jobs and consumes large number of system resources. Apache Spark’s in-memory capability at times comes a major roadblock for cost efficient processing of big data. Also, Spark does have its own file management system and hence needs to be integrated with other cloud based data platforms or apache hadoop.

* 1. **Question 51. Is It Necessary To Install Spark On All The Nodes Of A Yarn Cluster While Running Apache Spark On Yarn ?**

**Answer :**

No , it is not necessary because Apache Spark runs on top of YARN.

* 1. **Question 52. What Do You Understand By Executor Memory In A Spark Application?**

**Answer :**

Every spark application has same fixed heap size and fixed number of cores for a spark executor. The heap size is what referred to as the Spark executor memory which is controlled with the spark.executor.memory property of the –executor-memory flag. Every spark application will have one executor on each worker node. The executor memory is basically a measure on how much memory of the worker node will the application utilize.

* 1. **Question 53. What Does The Spark Engine Do?**

**Answer :**

Spark engine schedules, distributes and monitors the data application across the spark cluster.

* 1. **Question 54. What Makes Apache Spark Good At Low-latency Workloads Like Graph Processing And Machine Learning?**

**Answer :**

Apache Spark stores data in-memory for faster model building and training. Machine learning algorithms require multiple iterations to generate a resulting optimal model and similarly graph algorithms traverse all the nodes and edges.These low latency workloads that need multiple iterations can lead to increased performance. Less disk access and  controlled network traffic make a huge difference when there is lots of data to be processed.

* 1. **Question 55. Is It Necessary To Start Hadoop To Run Any Apache Spark Application ?**

**Answer :**

Starting hadoop is not manadatory to run any spark application. As there is no seperate storage in Apache Spark, it uses Hadoop HDFS but it is not mandatory. The data can be stored in local file system, can be loaded from local file system and processed.

* 1. **Question 56. What Is The Default Level Of Parallelism In Apache Spark?**

**Answer :**

If the user does not explicitly specify then the number of partitions are considered as default level of parallelism in Apache Spark.

* 1. **Question 57. Explain About The Common Workflow Of A Spark Program**

**Answer :**

* + 1. The foremost step in a Spark program involves creating input RDD's from external data.
    2. Use various RDD transformations like filter() to create new transformed RDD's based on the business logic.
    3. persist() any intermediate RDD's which might have to be reused in future.
    4. Launch various RDD actions() like first(), count() to begin parallel computation , which will then be optimized and executed by Spark.
  1. **Question 58. Name A Few Commonly Used Spark Ecosystems.**

**Answer :**

Spark SQL (Spark)

Spark Streaming

GraphX

MLlib

SparkR

* 1. **Question 59. What Is “spark Sql”?**

**Answer :**

Spark SQL is a Spark interface to work with structured as well as semi-structured data. It has the capability to load data from multiple structured sources like “text files”, JSON files, Parquet files, among others. Spark SQL provides a special type of RDD called SchemaRDD. These are row objects, where each object represents a record.

* 1. **Question 60. Can We Do Real-time Processing Using Spark Sql?**

**Answer :**

Not directly but we can register an existing RDD as a SQL table and trigger SQL queries on top of that.

* 1. **Question 61. What Is Spark Sql?**

**Answer :**

SQL Spark, better known as Spark 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.

* 1. **Question 62. What Is A Parquet File?**

**Answer :**

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.

* 1. **Question 63. List The Functions Of Spark Sql.**

**Answer :**

Spark SQL is capable of:

* + 1. Loading data from a variety of structured sources
    2. 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
    3. 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
  1. **Question 64. What Is Spark?**

**Answer :**

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

* 1. **Question 65. What Is Hive On Spark?**

**Answer :**

Hive is a component of Hortonworks’ Data Platform (HDP). Hive provides an SQL-like interface to data stored in the HDP. Spark users will automatically get the complete set of Hive’s rich features, including any new features that Hive might introduce in the future.

The main task around implementing the Spark execution engine for Hive lies in query planning, where Hive operator plans from the semantic analyzer which is translated to a task plan that Spark can execute. It also includes query execution, where the generated Spark plan gets actually executed in the Spark cluster.

* 1. **Question 66. What Is A “parquet” In Spark?**

**Answer :**

“Parquet” is a columnar format file supported by many data processing systems. Spark SQL performs both read and write operations with the “Parquet” file.

* 1. **Question 67. What Are Benefits Of Spark Over Mapreduce?**

**Answer :**

Due to the availability of in-memory processing, Spark implements the processing around 10-100x faster than Hadoop MapReduce. MapReduce makes use of persistence storage for any of the data processing tasks.

* + 1. Unlike Hadoop, Spark provides in-built libraries to perform multiple tasks form the same core like batch processing, Steaming, Machine learning, Interactive SQL queries. However, Hadoop only supports batch processing.
    2. Hadoop is highly disk-dependent whereas Spark promotes caching and in-memory data storage
    3. Spark is capable of performing computations multiple times on the same dataset. This is called iterative computation while there is no iterative computing implemented by Hadoop.
  1. **Question 68. How Sparksql Is Different From Hql And Sql?**

**Answer :**

SparkSQL is a special component on the spark Core engine that support SQL and Hive Query Language without changing any syntax. It’s possible to join SQL table and HQL table.

## ****Apache Spark Interview Questions And Answers****

### **1. Compare Hadoop and Spark.**

We will compare Hadoop MapReduce and Spark based on the following aspects:

|  |  |  |
| --- | --- | --- |
| **Apache Spark vs. Hadoop** | | |
| **Feature Criteria** | **Apache Spark** | **Hadoop** |
| **Speed** | 100 times faster than Hadoop | Decent speed |
| **Processing** | Real-time & Batch processing | Batch processing only |
| **Difficulty** | Easy because of high level modules | Tough to learn |
| **Recovery** | Allows recovery of partitions | Fault-tolerant |
| **Interactivity** | Has interactive modes | No interactive mode except Pig & Hive |

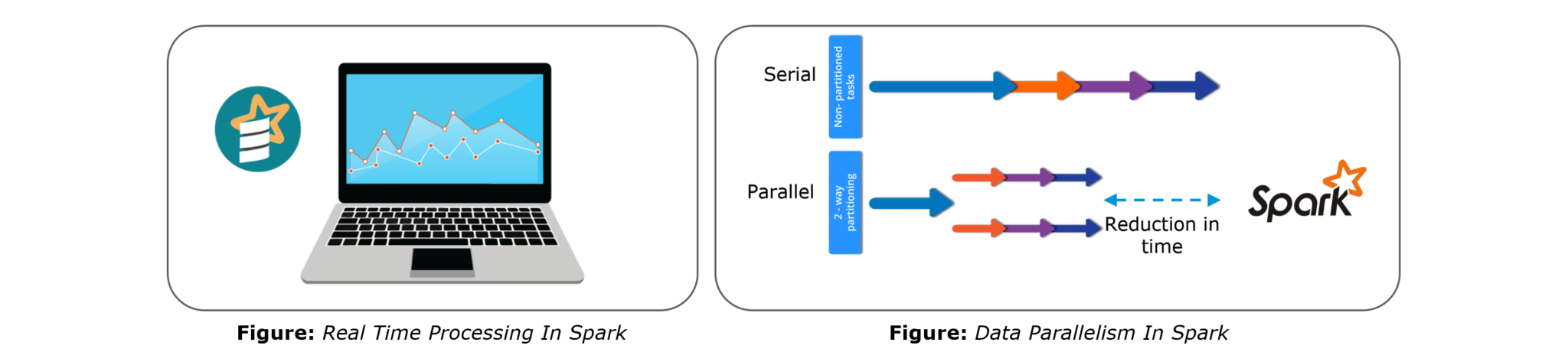
**Table:** Apache Spark versus Hadoop

Let us understand the same using an interesting analogy.

“Single cook cooking an entree is regular computing. Hadoop is multiple cooks cooking an entree into pieces and letting each cook her piece.Each cook has a separate stove and a food shelf. The first cook cooks the meat, the second cook cooks the sauce. This phase is called “Map”. A the end the main cook assembles the complete entree. This is called “Reduce”. For Hadoop, the cooks are not allowed to keep things on the stove between operations. Each time you make a particular operation, the cook puts results on the shelf. This slows things down.For Spark, the cooks are allowed to keep things on the stove between operations. This speeds things up. Finally, for Hadoop the recipes are written in a language which is illogical and hard to understand. For Spark, the recipes are nicely written.” – Stan Kladko*, Galactic Exchange.io*

### **2. What is Apache Spark?**

* [***Apache Spark***](https://www.edureka.co/blog/spark-tutorial/) is an open-source cluster computing framework for real-time processing.
* It has a thriving open-source community and is the most active Apache project at the moment.
* Spark provides an interface for programming entire clusters with implicit data parallelism and fault-tolerance.

Spark is of the most successful projects in the Apache Software Foundation. Spark has clearly evolved as the market leader for Big Data processing. Many organizations run Spark on clusters with thousands of nodes. Today, Spark is being adopted by major players like Amazon, eBay, and Yahoo!

### **3. Explain the key features of Apache Spark.**

The following are the key features of Apache Spark:

1. **Polyglot**
2. **Speed**
3. **Multiple Format Support**
4. **Lazy Evaluation**
5. **Real Time Computation**
6. **Hadoop Integration**
7. **Machine Learning**

Let us look at these features in detail:

1. **Polyglot**: Spark provides high-level APIs in Java, Scala, Python and R. Spark code can be written in any of these four languages. It provides a shell in Scala and Python. The Scala shell can be accessed through **./bin/spark-shell** and Python shell through **./bin/pyspark** from the installed directory.
2. **Speed**: Spark runs upto 100 times faster than Hadoop MapReduce for large-scale data processing. Spark is able to achieve this speed through controlled partitioning. It manages data using partitions that help parallelize distributed data processing with minimal network traffic.
3. **Multiple Formats**: Spark supports multiple data sources such as Parquet, JSON, Hive and Cassandra. The Data Sources API provides a pluggable mechanism for accessing structured data though Spark SQL. Data sources can be more than just simple pipes that convert data and pull it into Spark.
4. **Lazy Evaluation**: Apache Spark delays its evaluation till it is absolutely necessary. This is one of the key factors contributing to its speed. For transformations, Spark adds them to a DAG of computation and only when the driver requests some data, does this DAG actually gets executed.
5. **Real Time Computation**: Spark’s computation is real-time and has less latency because of its in-memory computation. Spark is designed for massive scalability and the Spark team has documented users of the system running production clusters with thousands of nodes and supports several computational models.
6. **Hadoop Integration**: Apache Spark provides smooth compatibility with Hadoop. This is a great boon for all the Big Data engineers who started their careers with Hadoop. Spark is a potential replacement for the MapReduce functions of Hadoop, while Spark has the ability to run on top of an existing Hadoop cluster using YARN for resource scheduling.
7. **Machine Learning**: Spark’s MLlib is the machine learning component which is handy when it comes to big data processing. It eradicates the need to use multiple tools, one for processing and one for machine learning. Spark provides data engineers and data scientists with a powerful, unified engine that is both fast and easy to use.

### **4. What are the languages supported by Apache Spark and which is the most popular one?**

Apache Spark supports the following four languages: Scala, Java, Python and R. Among these languages, Scala and Python have interactive shells for Spark. The Scala shell can be accessed through **./bin/spark-shell**and the Python shell through **./bin/pyspark**. Scala is the most used among them because Spark is written in Scala and it is the most popularly used for Spark.

### **5. What are benefits of Spark over MapReduce?**

Spark has the following benefits over MapReduce:

1. Due to the availability of in-memory processing, Spark implements the processing around 10 to 100 times faster than Hadoop MapReduce whereas MapReduce makes use of persistence storage for any of the data processing tasks.
2. Unlike Hadoop, Spark provides inbuilt libraries to perform multiple tasks from the same core like batch processing, Steaming, Machine learning, Interactive SQL queries. However, Hadoop only supports batch processing.
3. Hadoop is highly disk-dependent whereas Spark promotes caching and in-memory data storage.
4. Spark is capable of performing computations multiple times on the same dataset. This is called iterative computation while there is no iterative computing implemented by Hadoop.

### **6. 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. YARN is a distributed container manager, like Mesos for example, whereas Spark is a data processing tool. Spark can run on YARN, the same way Hadoop Map Reduce can run on YARN. Running Spark on YARN necessitates a binary distribution of Spark as built on YARN support.

### **7. Do you need to install Spark on all nodes of YARN cluster?**

No, because Spark runs on top of YARN. Spark runs independently from its installation. Spark has some options to use YARN when dispatching jobs to the cluster, rather than its own built-in manager, or Mesos. Further, there are some configurations to run YARN. They include master, deploy-mode, driver-memory, executor-memory, executor-cores, and queue.

### **8. Is there any benefit of learning MapReduce if Spark is better than MapReduce?**

Yes, MapReduce is a paradigm used by many big data tools including Spark as well. It is extremely relevant to use MapReduce when the data grows bigger and bigger. Most tools like Pig and Hive convert their queries into MapReduce phases to optimize them better.

### **9. Explain the concept of Resilient Distributed Dataset (RDD).**

RDD stands for Resilient Distribution Datasets. An RDD is a fault-tolerant collection of operational elements that run in parallel. The partitioned data in RDD is immutable and distributed in nature. There are primarily two types of RDD:

1. Parallelized Collections: Here, the existing RDDs running parallel with one another.
2. Hadoop Datasets: They perform functions on each file record in HDFS or other storage systems.

RDDs are basically parts of data that are stored in the memory distributed across many nodes. RDDs are lazily evaluated in Spark. This lazy evaluation is what contributes to Spark’s speed.

### **10. How do we create RDDs in Spark?**

Spark provides two methods to create RDD:

1. By parallelizing a collection in your Driver program.

2. This makes use of SparkContext’s ‘parallelize’

|  |  |
| --- | --- |
| 1  2  3 | method val DataArray = Array(2,4,6,8,10)    val DataRDD = sc.parallelize(DataArray) |

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

### **11. What is Executor Memory in a Spark application?**

Every spark application has same fixed heap size and fixed number of cores for a spark executor. The heap size is what referred to as the Spark executor memory which is controlled with the spark.executor.memory property of the **–executor-memory** flag. Every spark application will have one executor on each worker node. The executor memory is basically a measure on how much memory of the worker node will the application utilize.

### **12. Define Partitions in Apache Spark.**

As the name suggests, partition is a smaller and logical division of data similar to ‘split’ in MapReduce. It is a logical chunk of a large distributed data set. Partitioning is the process to derive logical units of data to speed up the processing process. Spark manages data using partitions that help parallelize distributed data processing with minimal network traffic for sending data between executors. By default, Spark tries to read data into an RDD from the nodes that are close to it. Since Spark usually accesses distributed partitioned data, to optimize transformation operations it creates partitions to hold the data chunks. Everything in Spark is a partitioned RDD.

### **13. What operations does RDD support?**

RDD (Resilient Distributed Dataset) is main logical data unit in Spark. An RDD has distributed a collection of objects. Distributed means, each RDD is divided into multiple partitions. Each of these partitions can reside in memory or stored on the disk of different machines in a cluster. RDDs are immutable (Read Only) data structure. You can’t change original RDD, but you can always transform it into different RDD with all changes you want.

RDDs support two types of operations: transformations and actions.

Transformations: Transformations create new RDD from existing RDD like map, reduceByKey and filter we just saw. Transformations are executed on demand. That means they are computed lazily.

Actions: Actions return final results of RDD computations. Actions triggers execution using lineage graph to load the data into original RDD, carry out all intermediate transformations and return final results to Driver program or write it out to file system.

### **14. What do you understand by Transformations in Spark?**

Transformations are functions applied on RDD, resulting into another RDD. It does not execute until an action occurs. map() and filter() are examples of transformations, where the former applies the function passed to it on each element of RDD and results into another RDD. The filter() creates a new RDD by selecting elements from current RDD that pass function argument.

|  |  |
| --- | --- |
| 1  2  3 | val rawData=sc.textFile("path to/movies.txt")    val moviesData=rawData.map(x=&gt;x.split("  ")) |

As we can see here, rawData RDD is transformed into moviesData RDD. Transformations are lazily evaluated.

### **15. Define Actions in Spark.**

An action helps in bringing back the data from RDD to the local machine. An action’s execution is the result of all previously created transformations. Actions triggers execution using lineage graph to load the data into original RDD, carry out all intermediate transformations and return final results to Driver program or write it out to file system.

reduce() is an action that implements the function passed again and again until one value if left. take() action takes all the values from RDD to a local node.

|  |  |
| --- | --- |
| 1 | moviesData.saveAsTextFile(“MoviesData.txt”) |

As we can see here, moviesData RDD is saved into a text file called MoviesData.txt.

### **16. Define functions of SparkCore.**

Spark Core is the base engine for large-scale parallel and distributed data processing. The core is the distributed execution engine and the Java, Scala, and Python APIs offer a platform for distributed ETL application development. SparkCore performs various important functions like memory management, monitoring jobs, fault-tolerance, job scheduling and interaction with storage systems. Further, additional libraries, built atop the core allow diverse workloads for streaming, SQL, and machine learning. It is responsible for:

1. Memory management and fault recovery
2. Scheduling, distributing and monitoring jobs on a cluster
3. Interacting with storage systems

### **17. What do you understand by Pair RDD?**

Apache defines PairRDD functions class as

|  |  |
| --- | --- |
| 1 | class PairRDDFunctions[K, V] extends Logging with HadoopMapReduceUtil with Serializable |

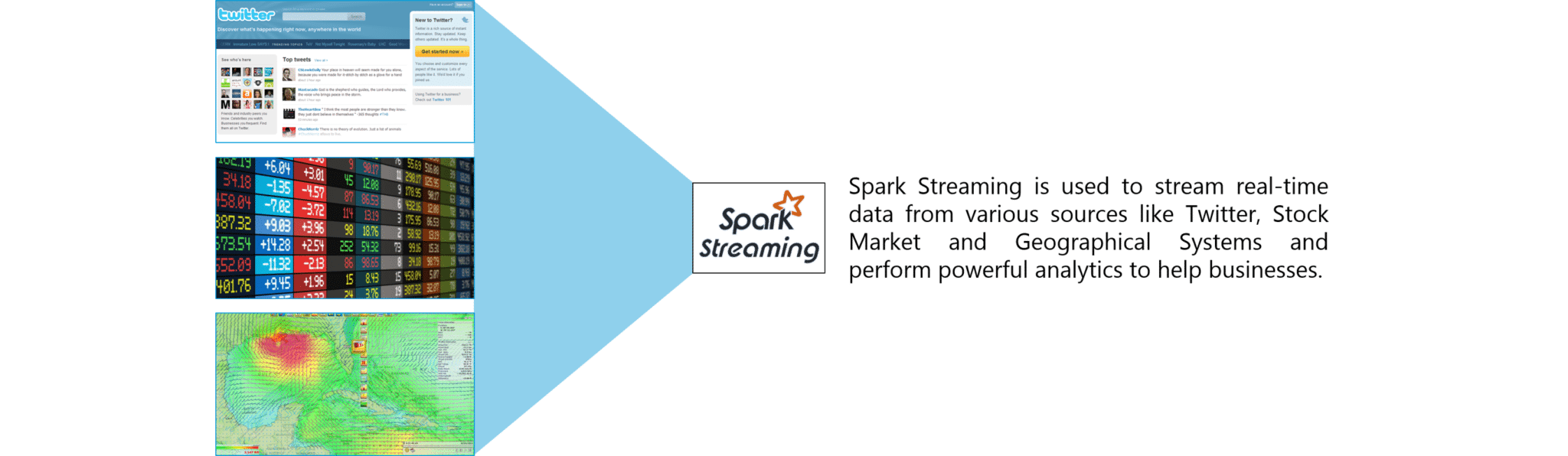
Special operations can be performed on RDDs in Spark using key/value pairs and such RDDs are referred to as Pair RDDs. Pair RDDs allow users to access each key in parallel. They have a reduceByKey() method that collects data based on each key and a join() method that combines different RDDs together, based on the elements having the same key.

### **18. Name the components of Spark Ecosystem.**

1. **Spark Core**: Base engine for large-scale parallel and distributed data processing
2. **Spark Streaming**: Used for processing real-time streaming data
3. **Spark SQL**: Integrates relational processing with Spark’s functional programming API
4. **GraphX**: Graphs and graph-parallel computation
5. **MLlib**: Performs machine learning in Apache Spark

### **19. How is Streaming implemented in Spark? Explain with examples.**

Spark Streaming is used for processing real-time streaming data. Thus it is a useful addition to the core Spark API. It enables high-throughput and fault-tolerant stream processing of live data streams. The fundamental stream unit is DStream which is basically a series of RDDs (Resilient Distributed Datasets) to process the real-time data. 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.

****

**Figure:** Spark Interview Questions – Spark Streaming

### **20. Is there an API for implementing graphs in Spark?**

GraphX is the Spark API for graphs and graph-parallel computation. Thus, it extends the Spark RDD with a Resilient Distributed Property Graph.

The property graph is a directed multi-graph which can have multiple edges in parallel. Every edge and vertex have user defined properties associated with it. Here, the parallel edges allow multiple relationships between the same vertices. At a high-level, GraphX extends the Spark RDD abstraction by introducing the Resilient Distributed Property Graph: a directed multigraph with properties attached to each vertex and edge.

To support graph computation, GraphX exposes a set of fundamental operators (e.g., subgraph, joinVertices, and mapReduceTriplets) as well as an optimized variant of the Pregel API. In addition, GraphX includes a growing collection of graph algorithms and builders to simplify graph analytics tasks.

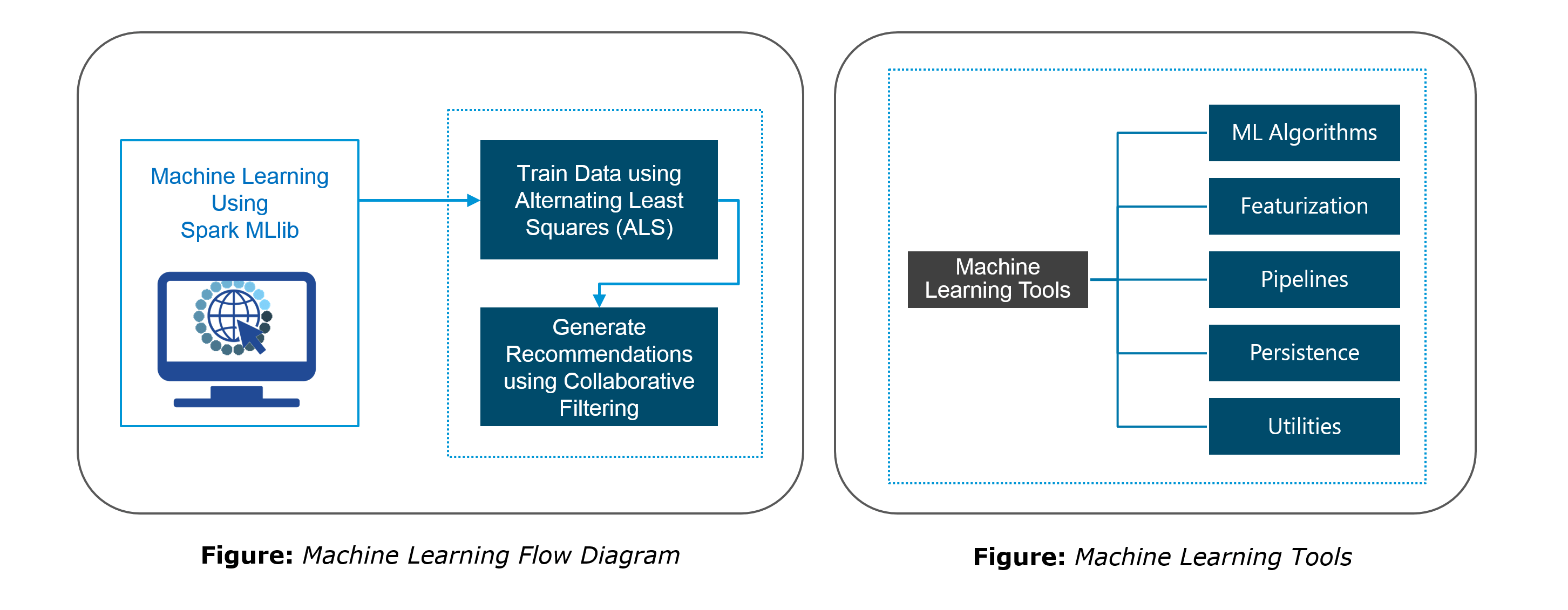
### **21. What is PageRank in GraphX?**

PageRank measures the importance of each vertex in a graph, assuming an edge from u to v represents an endorsement of v’s importance by u. For example, if a Twitter user is followed by many others, the user will be ranked highly.

GraphX comes with static and dynamic implementations of PageRank as methods on the PageRank Object. Static PageRank runs for a fixed number of iterations, while dynamic PageRank runs until the ranks converge (i.e., stop changing by more than a specified tolerance). GraphOps allows calling these algorithms directly as methods on Graph.

### **22. How is machine learning implemented in Spark?**

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.

****

### **23. Is there a module to implement SQL in Spark? How does it work?**

Spark SQL is a new module in Spark which integrates relational processing with Spark’s functional programming API. It supports querying data either via SQL or via the Hive Query Language. For those of you familiar with RDBMS, Spark SQL will be an easy transition from your earlier tools where you can extend the boundaries of traditional relational data processing.

Spark SQL integrates relational processing with Spark’s functional programming. Further, it provides support for various data sources and makes it possible to weave SQL queries with code transformations thus resulting in a very powerful tool.

The following are the four libraries of Spark SQL.

1. Data Source API
2. DataFrame API
3. Interpreter & Optimizer
4. SQL Service

### Spark SQL - Spark Interview Questions - Edureka**24. What is a Parquet file?**

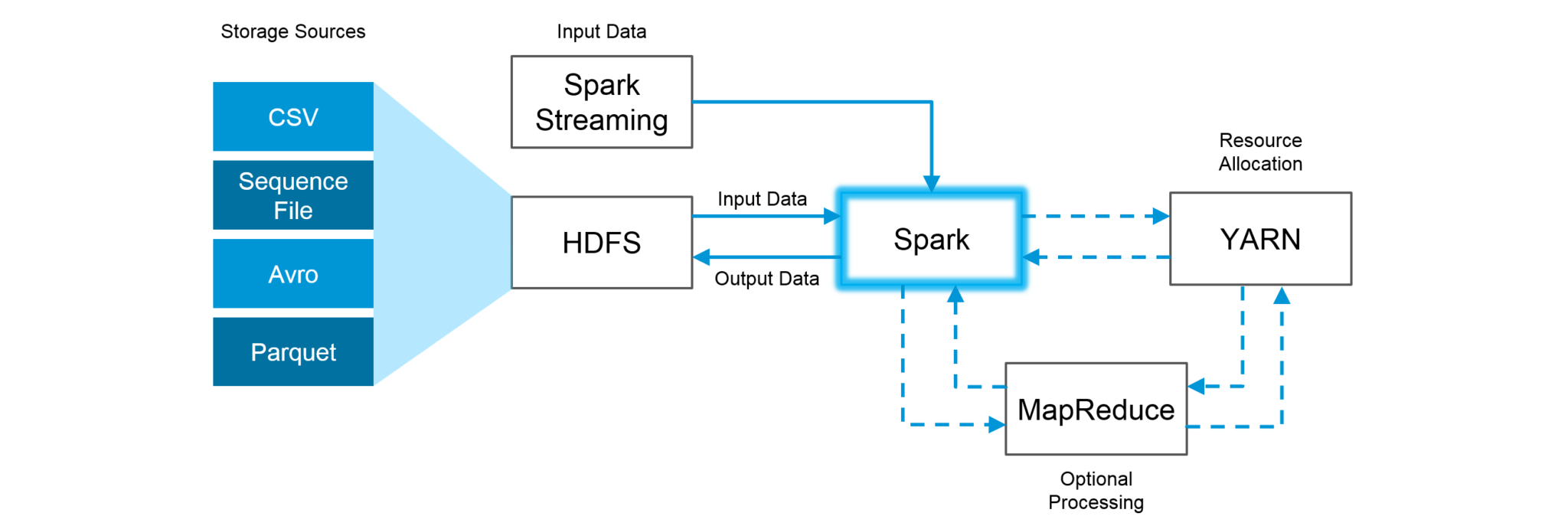
Parquet is a columnar format file supported by many other data processing systems. Spark SQL performs both read and write operations with Parquet file and consider it be one of the best big data analytics formats so far.

Parquet is a columnar format, supported by many data processing systems. The advantages of having a columnar storage are as follows:

1. Columnar storage limits IO operations.
2. It can fetch specific columns that you need to access.
3. Columnar storage consumes less space.
4. It gives better-summarized data and follows type-specific encoding.

### **25. How can Apache Spark be used alongside Hadoop?**

The best part of Apache Spark is its compatibility with Hadoop. As a result, this makes for a very powerful combination of technologies. Here, we will be looking at how Spark can benefit from the best of Hadoop. Using Spark and Hadoop together helps us to leverage Spark’s processing to utilize the best of Hadoop’s HDFS and YARN.

**Figure:**Using Spark and Hadoop

Hadoop components can be used alongside Spark in the following ways:

1. **HDFS**: Spark can run on top of HDFS to leverage the distributed replicated storage.
2. **MapReduce**: Spark can be used along with MapReduce in the same Hadoop cluster or separately as a processing framework.
3. **YARN**: Spark applications can also be run on YARN (Hadoop NextGen).
4. **Batch & Real Time Processing**: MapReduce and Spark are used together where MapReduce is used for batch processing and Spark for real-time processing.

### **26. What is RDD Lineage?**

Spark does not support data replication in the memory and thus, if any data is lost, it is rebuild using RDD lineage. RDD lineage is a process that reconstructs lost data partitions. The best is that RDD always remembers how to build from other datasets.

### **27. 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, a 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.

### **28. What file systems does Spark support?**

The following three file systems are supported by Spark:

1. Hadoop Distributed File System (HDFS).
2. Local File system.
3. Amazon S3

### **29. List the functions of Spark SQL.**

Spark SQL is capable of:

1. Loading data from a variety of structured sources.
2. 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.
3. 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.

### **30. What is Spark Executor?**

When SparkContext connects 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.

### **31. Name types of Cluster Managers in Spark.**

The Spark framework supports three major types of Cluster Managers:

1. **Standalone**: A basic manager to set up a cluster.
2. **Apache Mesos**: Generalized/commonly-used cluster manager, also runs Hadoop MapReduce and other applications.
3. **YARN**: Responsible for resource management in Hadoop.

### **32. What do you understand by worker node?**

Worker node refers to any node that can run the application code in a cluster. The driver program must listen for and accept incoming connections from its executors and must be network addressable from the worker nodes.

Worker node is basically the slave node. Master node assigns work and worker node actually performs the assigned tasks. Worker nodes process the data stored on the node and report the resources to the master. Based on the resource availability, the master schedule tasks.

### **33. Illustrate some demerits of using Spark.**

The following are some of the demerits of using Apache Spark:

1. Since Spark utilizes more storage space compared to Hadoop and MapReduce, there may arise certain problems.
2. Developers need to be careful while running their applications in Spark.
3. Instead of running everything on a single node, the work must be distributed over multiple clusters.
4. Spark’s “in-memory” capability can become a bottleneck when it comes to cost-efficient processing of big data.
5. Spark consumes a huge amount of data when compared to Hadoop.

### **34. List some use cases where Spark outperforms Hadoop in processing.**

1. **Sensor Data Processing**: Apache Spark’s “In-memory” computing works best here, as data is retrieved and combined from different sources.
2. **Real Time Processing**: Spark is preferred over Hadoop for real-time querying of data. e.g. Stock Market Analysis, Banking, Healthcare, Telecommunications, etc.
3. **Stream Processing**: For processing logs and detecting frauds in live streams for alerts, Apache Spark is the best solution.
4. **Big Data Processing**:Spark runs upto 100 times faster than Hadoop when it comes to processing medium and large-sized datasets.

### **35. What is a Sparse Vector?**

A sparse vector has two parallel arrays; one for indices and the other for values. These vectors are used for storing non-zero entries to save space.

|  |  |
| --- | --- |
| 1 | Vectors.sparse(7,Array(0,1,2,3,4,5,6),Array(1650d,50000d,800d,3.0,3.0,2009,95054)) |

The above sparse vector can be used instead of dense vectors.

|  |  |
| --- | --- |
| 1 | val myHouse = Vectors.dense(4450d,2600000d,4000d,4.0,4.0,1978.0,95070d,1.0,1.0,1.0,0.0) |

### **36. Can you use Spark to access and analyze data stored in Cassandra databases?**

Yes, it is possible if you use Spark Cassandra Connector.To connect Spark to a Cassandra cluster, a Cassandra Connector will need to be added to the Spark project. In the setup, a Spark executor will talk to a local Cassandra node and will only query for local data. It makes queries faster by reducing the usage of the network to send data between Spark executors (to process data) and Cassandra nodes (where data lives).

### **37. Is it possible to run Apache Spark on Apache Mesos?**

Yes, Apache Spark can be run on the hardware clusters managed by Mesos. In a standalone cluster deployment, the cluster manager in the below diagram is a Spark master instance. When using Mesos, the Mesos master replaces the Spark master as the cluster manager. Mesos determines what machines handle what tasks. Because it takes into account other frameworks when scheduling these many short-lived tasks, multiple frameworks can coexist on the same cluster without resorting to a static partitioning of resources.

### **38. How can Spark be connected to Apache Mesos?**

To connect Spark with Mesos:

1. Configure the spark driver program to connect to Mesos.
2. Spark binary package should be in a location accessible by Mesos.
3. Install Apache Spark in the same location as that of Apache Mesos and configure the property ‘spark.mesos.executor.home’ to point to the location where it is installed.

### **39. How can you minimize data transfers when working with Spark?**

Minimizing data transfers and avoiding shuffling helps write spark programs that run in a fast and reliable manner. The various ways in which data transfers can be minimized when working with Apache Spark are:

1. Using Broadcast Variable- Broadcast variable enhances the efficiency of joins between small and large RDDs.
2. Using Accumulators – Accumulators help update the values of variables in parallel while executing.

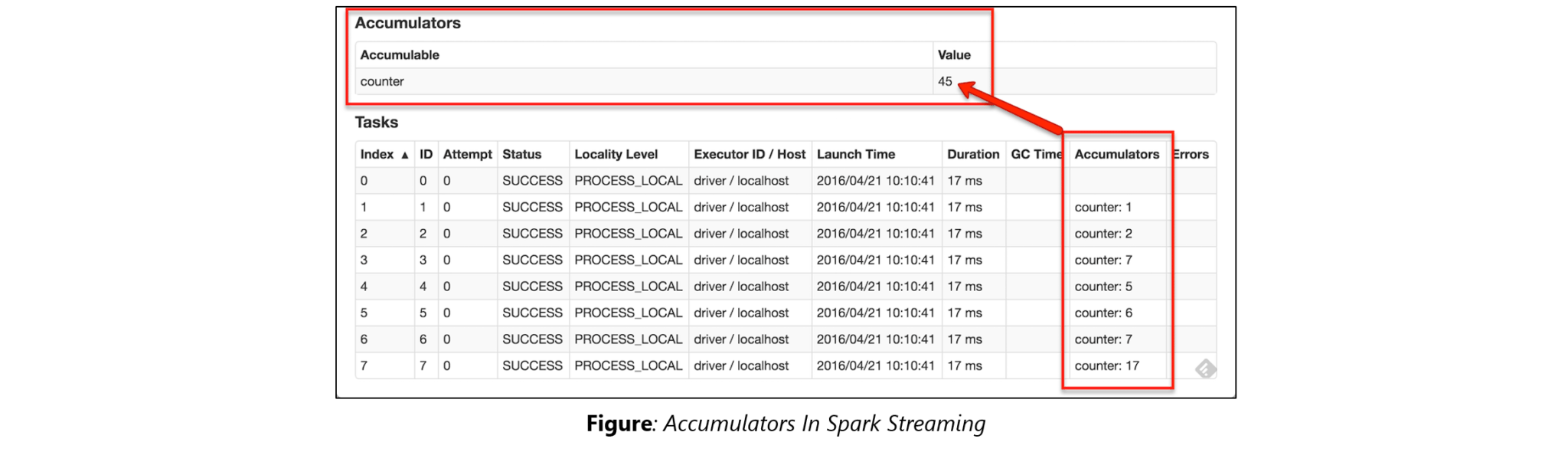
The most common way is to avoid operations ByKey, repartition or any other operations which trigger shuffles.

### **40. What are broadcast variables?**

Broadcast variables allow the programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks. They can be used to give every node a copy of a large input dataset in an efficient manner. Spark also attempts to distribute broadcast variables using efficient broadcast algorithms to reduce communication cost.

### Broadcast Variables - Spark Interview Questions - Edureka**41. Explain accumulators in Apache Spark.**

Accumulators are variables that are only added through an associative and commutative operation. They are used to implement counters or sums. Tracking accumulators in the UI can be useful for understanding the progress of running stages. Spark natively supports numeric accumulators. We can create named or unnamed accumulators.

****

### **42. Why is there a need for broadcast variables when working with Apache Spark?**

Broadcast variables are read only variables, present in-memory cache on every machine. When working with Spark, usage of broadcast variables eliminates the necessity to ship copies of a variable for every task, so data can be processed faster. Broadcast variables help in storing a lookup table inside the memory which enhances the retrieval efficiency when compared to an RDD lookup().

### **43. How can you trigger automatic clean-ups in Spark to handle accumulated metadata?**

You can trigger the clean-ups by setting the parameter ‘spark.cleaner.ttl’ or by dividing the long running jobs into different batches and writing the intermediary results to the disk.

### **44. What is the significance of Sliding Window operation?**

Sliding Window controls transmission of data packets between various computer networks. Spark Streaming library provides windowed computations where the transformations on RDDs are applied over a sliding window of data. Whenever the window slides, the RDDs that fall within the particular window are combined and operated upon to produce new RDDs of the windowed DStream.

### DStream Sliding Window - Spark Interview Questions - Edureka**45. What is a DStream in Apache Spark?**

***Discretized Stream***(DStream) is the basic abstraction provided by Spark Streaming. It is a continuous stream of data. It is received from a data source or from a processed data stream generated by transforming the input stream. Internally, a DStream is represented by a continuous series of RDDs and each RDD contains data from a certain interval. Any operation applied on a DStream translates to operations on the underlying RDDs.

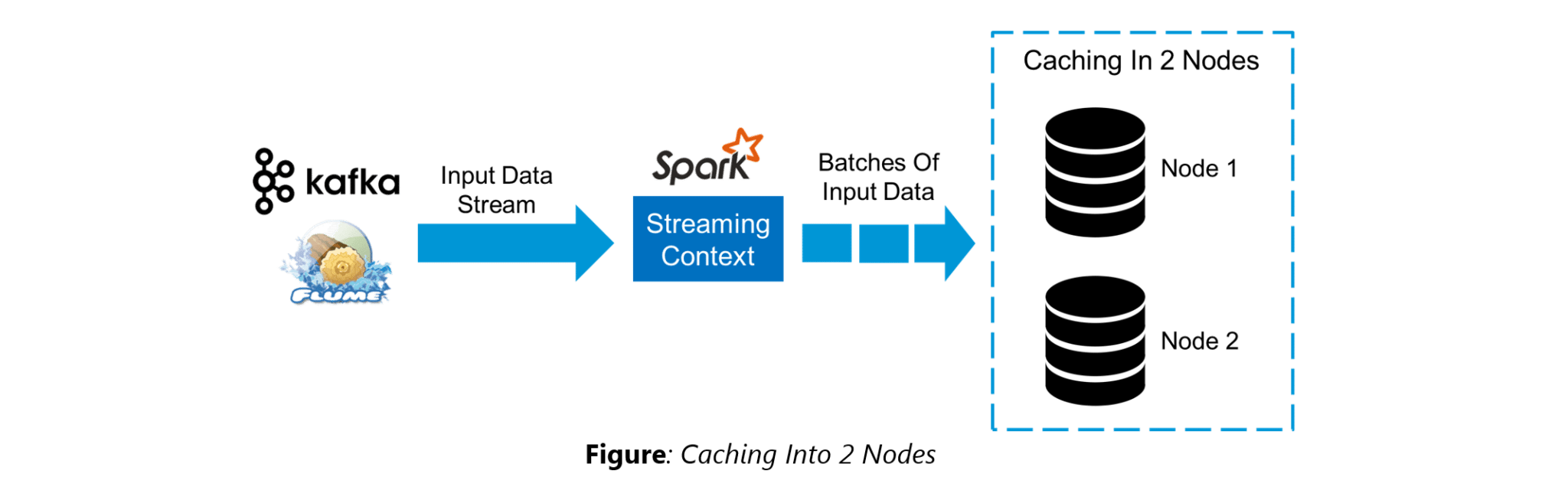
DStreams can be created from various sources like Apache Kafka, HDFS, and Apache Flume. DStreams have two operations:

1. Transformations that produce a new DStream.
2. Output operations that write data to an external system.

There are many DStream transformations possible in Spark Streaming. Let us look at **filter(*func*)**. filter(*func*) returns a new DStream by selecting only the records of the source DStream on which func returns true.

### DStream Filter - Spark Interview Questions - Edureka**46. Explain Caching in Spark Streaming.**

DStreams allow developers to cache/ persist the stream’s data in memory. This is useful if the data in the DStream will be computed multiple times. This can be done using the persist() method on a DStream. For input streams that receive data over the network (such as Kafka, Flume, Sockets, etc.), the default persistence level is set to replicate the data to two nodes for fault-tolerance.

****

### **47. When running Spark applications, is it necessary to install Spark on all the nodes of YARN cluster?**

Spark need not be installed when running a job under YARN or Mesos because Spark can execute on top of YARN or Mesos clusters without affecting any change to the cluster.

### **48. What are the various data sources available in Spark SQL?**

Parquet file, JSON datasets and Hive tables are the data sources available in Spark SQL.

### **49. What are the various levels of persistence in Apache Spark?**

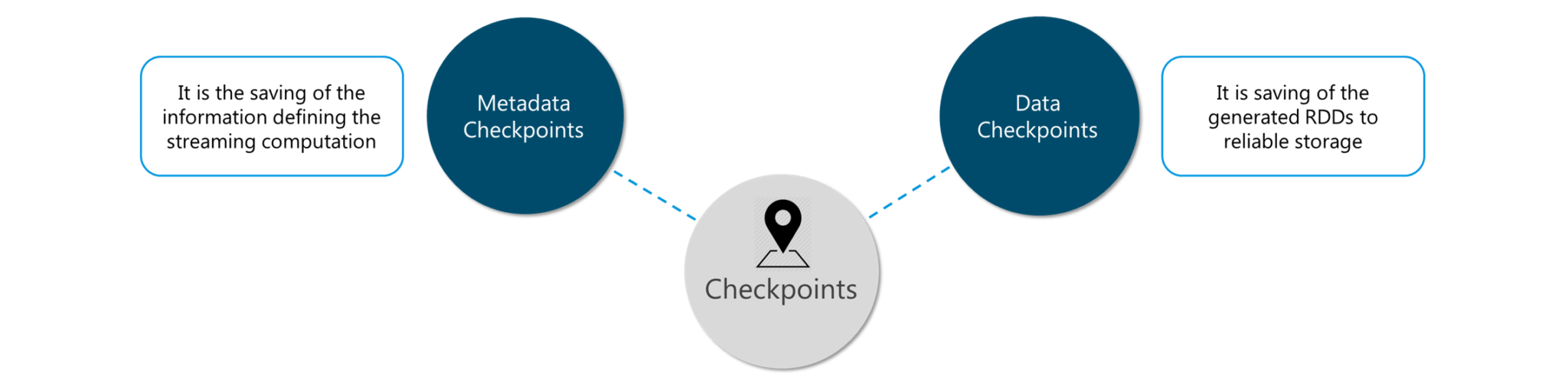
Apache Spark automatically persists the intermediary data from various shuffle operations, however, it is often suggested that users call persist () method on the RDD in case they plan to reuse it. Spark has various persistence levels to store the RDDs on disk or in memory or as a combination of both with different replication levels.

The various storage/persistence levels in Spark are:

1. MEMORY\_ONLY: Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, some partitions will not be cached and will be recomputed on the fly each time they’re needed. This is the default level.
2. MEMORY\_AND\_DISK: Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, store the partitions that don’t fit on disk, and read them from there when they’re needed.
3. MEMORY\_ONLY\_SER: Store RDD as *serialized* Java objects (one byte array per partition).
4. MEMORY\_AND\_DISK\_SER: Similar to MEMORY\_ONLY\_SER, but spill partitions that don’t fit in memory to disk instead of recomputing them on the fly each time they’re needed.
5. DISK\_ONLY: Store the RDD partitions only on disk.
6. OFF\_HEAP: Similar to MEMORY\_ONLY\_SER, but store the data in off-heap memory.

### **50. Does Apache Spark provide checkpoints?**

Checkpoints are similar to checkpoints in gaming. They make it run 24/7 and make it resilient to failures unrelated to the application logic.

**Figure:** Spark Interview Questions – Checkpoints

Lineage graphs are always useful to recover RDDs from a failure but this is generally time-consuming if the RDDs have long lineage chains. Spark has an API for checkpointing i.e. a REPLICATE flag to persist. However, the decision on which data to checkpoint – is decided by the user. Checkpoints are useful when the lineage graphs are long and have wide dependencies.

### **51. How Spark uses Akka?**

Spark uses Akka basically for scheduling. All the workers request for a task to master after registering. The master just assigns the task. Here Spark uses Akka for messaging between the workers and masters.

### **52. What do you understand by Lazy Evaluation?**

Spark is intellectual in the manner in which it operates on data. When you tell Spark to operate on a given dataset, it heeds the instructions and makes a note of it, so that it does not forget – but it does nothing, unless asked for the final result. When a transformation like map() is called on an RDD, the operation is not performed immediately. Transformations in Spark are not evaluated till you perform an action. This helps optimize the overall data processing workflow.

### Lazy Evaluation - Spark Interview Questions - Edureka**53. What do you understand by SchemaRDD in Apache Spark RDD?**

SchemaRDD is an RDD that consists of row objects (wrappers around the basic string or integer arrays) with schema information about the type of data in each column.

SchemaRDD was designed as an attempt to make life easier for developers in their daily routines of code debugging and unit testing on SparkSQL core module. The idea can boil down to describing the data structures inside RDD using a formal description similar to the relational database schema. On top of all basic functions provided by common RDD APIs, SchemaRDD also provides some straightforward relational query interface functions that are realized through SparkSQL.

Now, it is officially renamed to DataFrame API on Spark’s latest trunk.

### **54. How is Spark SQL different from HQL and SQL?**

Spark SQL is a special component on the Spark Core engine that supports SQL and Hive Query Language without changing any syntax. It is possible to join SQL table and HQL table to Spark SQL.

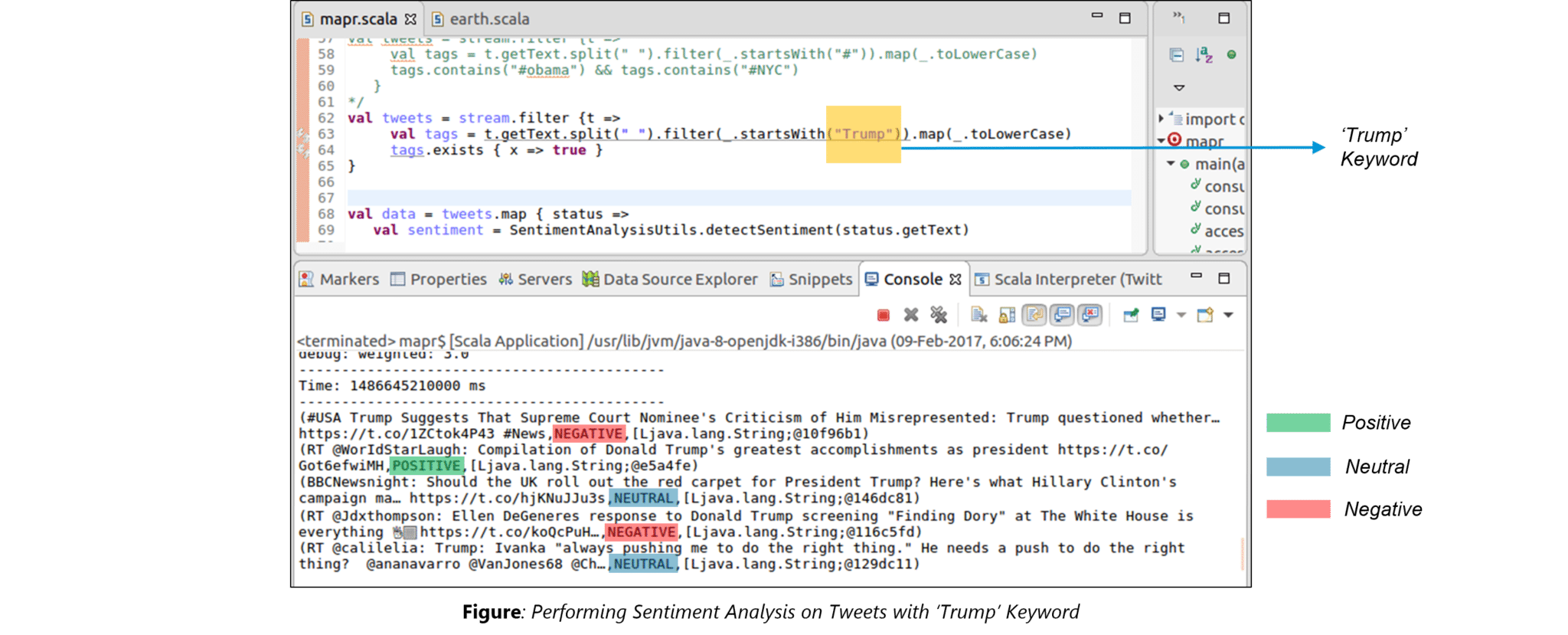
### **55. Explain a scenario where you will be using Spark Streaming.**

When it comes to Spark Streaming, the data is streamed in real-time onto our Spark program.

Twitter Sentiment Analysis is a real-life use case of Spark Streaming. Trending Topics can be used to create campaigns and attract a larger audience. It helps in crisis management, service adjusting and target marketing.

Sentiment refers to the emotion behind a social media mention online. Sentiment Analysis is categorizing the tweets related to a particular topic and performing data mining using Sentiment Automation Analytics Tools.

Spark Streaming can be used to gather live tweets from around the world into the Spark program. This stream can be filtered using Spark SQL and then we can filter tweets based on the sentiment. The filtering logic will be implemented using MLlib where we can learn from the emotions of the public and change our filtering scale accordingly.



The above figure displays the sentiments for the tweets containing the word ‘Trump’.

### **Compare MapReduce with Spark.**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **MapReduce** | **Spark** |
| Processing speed | Good | Excellent (up to 100 times faster) |
| Data caching | Hard disk | In-memory |
| Performing iterative jobs | Average | Excellent |
| Dependency on Hadoop | Yes | No |
| Machine Learning applications | Average | Excellent |

### **2. What is Apache Spark?**

Spark is a fast, easy-to-use, and flexible data processing framework. It has an advanced execution engine supporting a cyclic data flow and in-memory computing. [Apache Spark](https://intellipaat.com/blog/what-is-apache-spark/) can run standalone, on Hadoop, or in the cloud and is capable of accessing diverse data sources including HDFS, HBase, and Cassandra, among others.

### **3. Explain the key features of Spark.**

* Apache Spark allows integrating with Hadoop.
* It has an interactive language shell, Scala (the language in which Spark is written).
* Spark consists of RDDs (Resilient Distributed Datasets), which can be cached across the computing nodes in a cluster.
* Apache Spark supports multiple analytic tools that are used for interactive query analysis, real-time analysis, and graph processing

### **4. Define RDD.**

RDD is the acronym for Resilient Distribution Datasets—a fault-tolerant collection of operational elements that run in parallel. The partitioned data in an RDD is immutable and distributed. There are primarily two types of RDDs:

* **Parallelized collections:** The existing RDDs running in parallel with one another
* **Hadoop datasets:** Those performing a function on each file record in HDFS or any other storage system

### **5. What does a Spark Engine do?**

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

***Read on Spark Engine and more in this***[***Apache Spark Community***](https://intellipaat.com/community/big-data-hadoop-spark)***!***

### **6. Define Partitions.**

As the name suggests, a partition is a smaller and logical division of data similar to a ‘split’ in MapReduce. Partitioning is the process of deriving logical units of data to speed up data processing. Everything in Spark is a partitioned RDD.

### **7. What operations does an RDD support?**

* Transformations
* Actions

### **8. What do you understand by Transformations in Spark?**

Transformations are functions applied to RDDs, resulting in another RDD. It does not execute until an action occurs. Functions such as map() and filer() are examples of transformations, where the map() function iterates over every line in the RDD and splits into a new RDD. The filter() function creates a new RDD by selecting elements from the current RDD that passes the function argument.

### **9. Define Actions in Spark.**

In Spark, an action helps in bringing back data from an RDD to the local machine. They are RDD operations giving non-RDD values. The reduce() function is an action that is implemented again and again until only one value if left. The take() action takes all the values from an RDD to the local node.

### **10. Define the functions of Spark Core.**

Serving as the base engine, Spark Core performs various important functions like memory management, monitoring jobs, providing fault-tolerance, job scheduling, and interaction with storage systems.

### **11. What is RDD Lineage?**

Spark does not support data replication in memory and thus, if any data is lost, it is rebuild using RDD lineage. RDD lineage is a process that reconstructs lost data partitions. The best thing about this is that RDDs always remember how to build from other datasets.

### **12. What is Spark Driver?**

Spark driver is the program that runs on the master node of a machine and declares transformations and actions on data RDDs. In simple terms, a driver in Spark creates SparkContext, connected to a given Spark Master. It also delivers RDD graphs to Master, where the standalone Cluster Manager runs.

### **13. 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 supports Spark on YARN mode by default.

### **14. Name the commonly used Spark Ecosystems.**

* Spark SQL (Spark) for developers
* Spark Streaming for processing live data streams
* GraphX for generating and computing graphs
* MLlib (Machine Learning Algorithms)
* SparkR to promote R programming in the Spark engine

### **15. Define Spark Streaming.**

Spark supports stream processing—an extension to the Spark API allowing stream processing of live data streams. Data from different sources like Kafka, Flume, Kinesis is processed and then pushed to file systems, live dashboards, and databases. It is similar to batch processing in terms of the input data which is here divided into streams like batches in batch processing.

***Learn in detail about the***[***Top Four Apache Spark Use Cases***](https://intellipaat.com/blog/top-4-apache-spark-use-cases/)***including Spark Streaming!***

### **16. What is GraphX?**

Spark uses GraphX for graph processing to build and transform interactive graphs. The GraphX component enables programmers to reason about structured data at scale.

### **17. What does MLlib do?**

MLlib is a 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 the like.

### **18. What is Spark SQL?**

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

### **19. 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 the Parquet file and considers it be one of the best Big Data Analytics formats so far.

### **20. What file systems does Apache Spark support?**

* Hadoop Distributed File System (HDFS)
* Local file system
* Amazon S3

### **21. 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 needs a binary distribution of Spark that is built on YARN support.

### **22. 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), e.g., using Business Intelligence tools like Tableau
* Providing rich integration between SQL and the regular Python/Java/Scala code, including the ability to join RDDs and SQL tables, expose custom functions in SQL, and more.

### **23. What are the benefits of Spark over MapReduce?**

* Due to the availability of in-memory processing, Spark implements data processing 10–100x faster than Hadoop MapReduce. MapReduce, on the other hand, makes use of persistence storage for any of the data processing tasks.
* Unlike Hadoop, Spark provides in-built libraries to perform multiple tasks using batch processing, steaming, Machine Learning, and interactive SQL queries. However, Hadoop only supports batch processing.
* Hadoop is highly disk-dependent, whereas Spark promotes caching and in-memory data storage.
* Spark is capable of performing computations multiple times on the same dataset, which is called iterative computation. Whereas, there is no iterative computing implemented by Hadoop.

### **24. Is there any benefit of learning MapReduce?**

Yes, MapReduce is a paradigm used by many Big Data tools, including Apache Spark. It becomes extremely relevant to use MapReduce when data grows bigger and bigger. Most tools like Pig and Hive convert their queries into MapReduce phases to optimize them better.

### **25. What is Spark Executor?**

When SparkContext connects to Cluster Manager, it acquires an executor on the nodes in the cluster. Executors are Spark processes that run computations and store data on worker nodes. The final tasks by SparkContext are transferred to executors for their execution.

### **26. Name the types of Cluster Managers in Spark.**

The Spark framework supports three major types of Cluster Managers.

* **Standalone:** A basic Cluster Manager to set up a cluster
* **Apache Mesos:** A generalized/commonly-used Cluster Manager, running Hadoop MapReduce and other applications
* **YARN:** A Cluster Manager responsible for resource management in Hadoop

### **27. What do you understand by a Worker node?**

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

### **28. What is PageRank?**

A unique feature and algorithm in GraphX, PageRank is the measure of each vertex in a graph. For instance, an edge from u to v represents an endorsement of v‘s importance w.r.t. u. In simple terms, if a user at Instagram is followed massively, he/she will be ranked high on that platform.

**Interested in learning Spark? Take up our**[***Spark Training in Sydney***](https://intellipaat.com/apache-spark-scala-training-sydney/)**now!**

### **29. Do you need to install Spark on all the nodes of the YARN cluster while running Spark on YARN?**

No, because Spark runs on top of YARN.

### **30. Illustrate some demerits of using Spark.**

Since Spark utilizes more storage space when compared to Hadoop and MapReduce, there might arise certain problems. Developers need to be careful while running their applications on Spark. To resolve the issue, they can think of distributing the workload over multiple clusters, instead of running everything on a single node.

### **31. How to create an RDD?**

Spark provides two methods to create an RDD:

* By parallelizing a collection in the driver program. This makes use of SparkContext’s ‘parallelize’ method **val**

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

val distIntellipaatData = sc.parallelize(IntellipaatData)

# Apache Spark Architecture – Spark Cluster Architecture Explained

Apache Spark is an open-source cluster computing framework which is setting the world of Big Data on fire. According to [***Spark Certified Experts***](https://www.edureka.co/apache-spark-scala-training), Sparks performance is up to 100 times faster in memory and 10 times faster on disk when compared to Hadoop. In this blog, I will give you a brief insight on Spark Architecture and the fundamentals that underlie Spark Architecture.

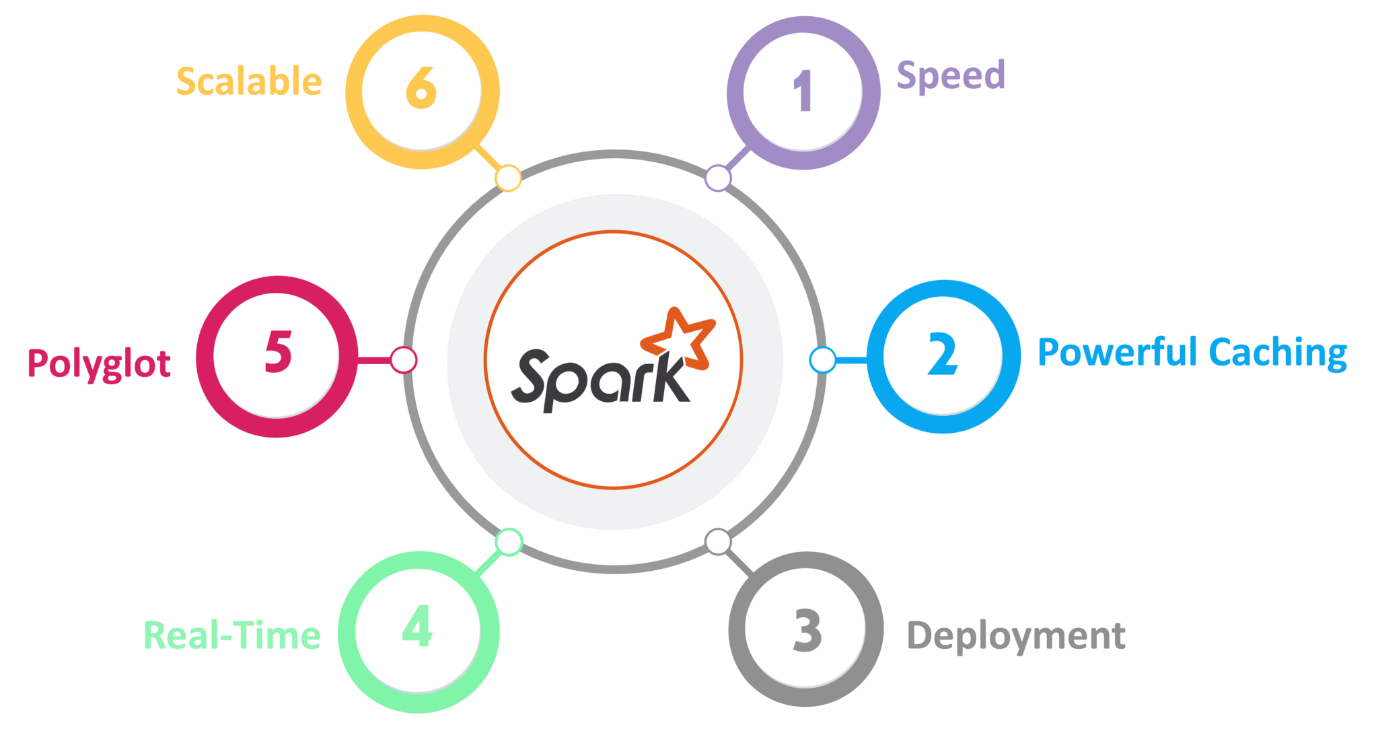
In this Spark Architecture article, I will be covering the following topics:

* [Spark & its Features](https://www.edureka.co/blog/spark-architecture/#Spark)
* [Spark Architecture Overview](https://www.edureka.co/blog/spark-architecture/#Overview)
* [Spark Eco-System](https://www.edureka.co/blog/spark-architecture/#SparkE)
* [Resilient Distributed Datasets (RDDs)](https://www.edureka.co/blog/spark-architecture/#rdds)
* [Working of Spark Architecture](https://www.edureka.co/blog/spark-architecture/#Working)
* [Example using Scala in Spark Shell](https://www.edureka.co/blog/spark-architecture/#Scala)

## ****Spark & its Features****

Apache Spark is an open source cluster computing framework for real-time data processing. The main feature of Apache Spark is its ***in-memory cluster computing*** that increases the processing speed of an application. Spark provides an interface for programming entire clusters with implicit **data parallelism and fault tolerance**. It is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries, and streaming.

### **Features of Apache Spark:**

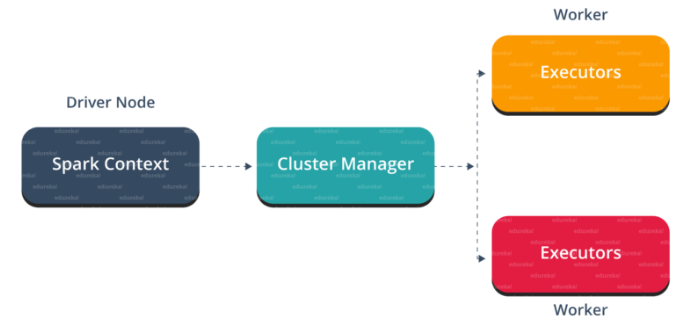
                                                           Fig: Features of Spark

1. **Speed**Spark runs up to 100 times faster than Hadoop MapReduce for large-scale data processing. It is also able to achieve this speed through controlled partitioning.
2. **Powerful Caching**Simple programming layer provides powerful caching and disk persistence capabilities.
3. **Deployment**It can be deployed through **Mesos, Hadoop via YARN, or Spark’s own cluster manager.**
4. **Real-Time**  
   It offers Real-time computation & low latency because of **in-memory computation.**
5. **Polyglot**  
   Spark provides high-level APIs in Java, Scala, Python, and R. Spark code can be written in any of these four languages. It also provides a shell in Scala and Python.

## ****Spark Architecture Overview****

Apache Spark has a well-defined layered architecture where all the spark components and layers are loosely coupled. This architecture is further integrated with various extensions and libraries. Apache Spark Architecture is based on two main abstractions:

* Resilient Distributed Dataset (RDD)
* Directed Acyclic Graph (DAG)

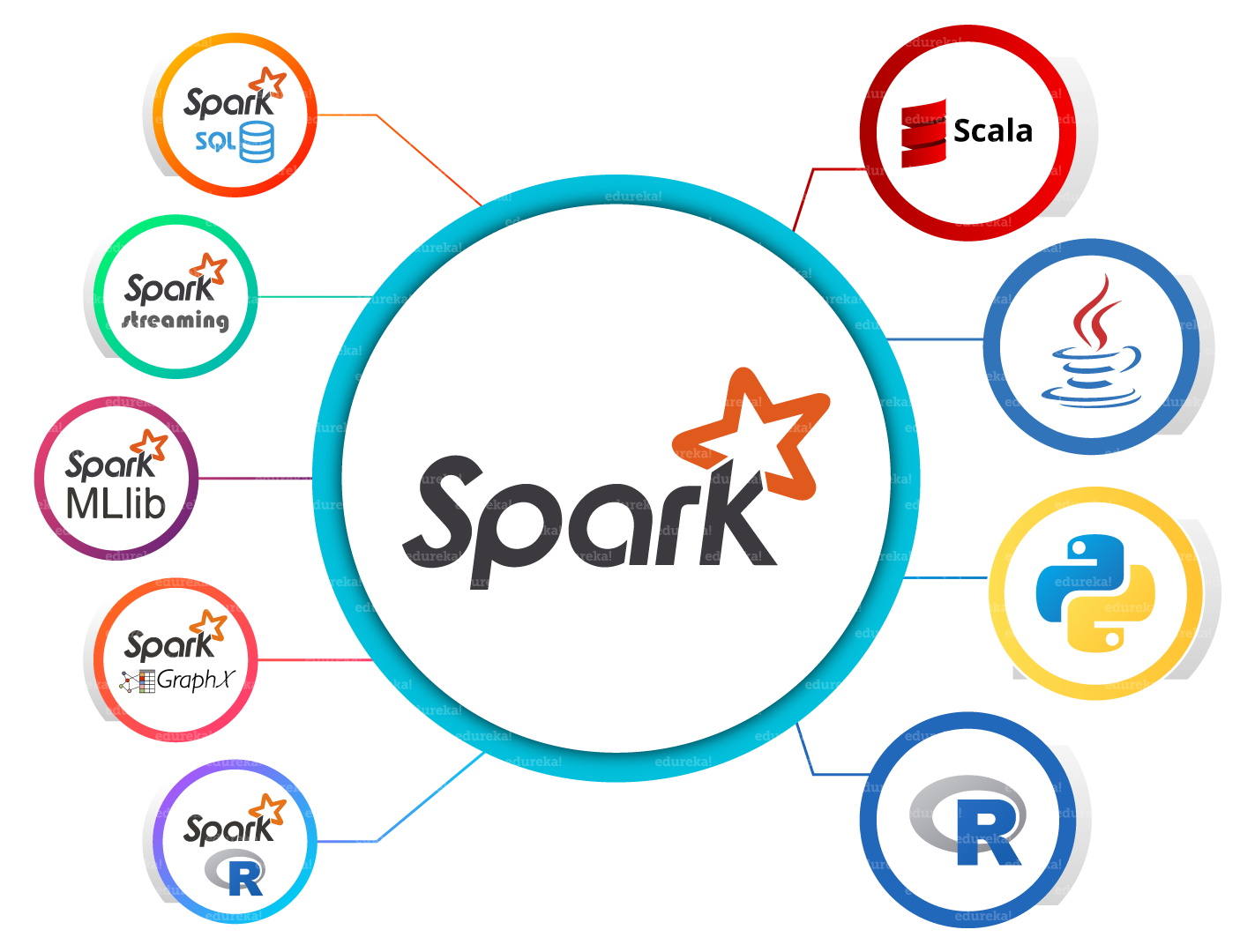
                                                        Fig: Spark Architecture

But before diving any deeper into the Spark architecture, let me explain few fundamental concepts of Spark like Spark Eco-system and RDD. This will help you in gaining better insights.

Let me first explain what is Spark Eco-System.

## ****Spark Eco-System****

As you can see from the below image, the spark ecosystem is composed of various components like Spark SQL, Spark Streaming, MLlib, GraphX, and the Core API component.

                                                              Fig: Spark Eco-System

1. **SparkCore**Spark Core is the base engine for large-scale parallel and distributed data processing. Further, additional libraries which are built on the top of the core allows diverse workloads for streaming, SQL, and machine learning. It is responsible for memory management and fault recovery, scheduling, distributing and monitoring jobs on a cluster & interacting with storage systems.
2. **SparkStreaming**Spark Streaming is the component of Spark which is used to process real-time streaming data. Thus, it is a useful addition to the core Spark API. It enables high-throughput and fault-tolerant stream processing of live data streams.
3. **SparkSQL**Spark SQL is a new module in Spark which integrates relational processing with Spark’s functional programming API. It supports querying data either via SQL or via the Hive Query Language. For those of you familiar with RDBMS, Spark SQL will be an easy transition from your earlier tools where you can extend the boundaries of traditional relational data processing.
4. **GraphX**GraphX is the Spark API for graphs and graph-parallel computation. Thus, it extends the Spark RDD with a Resilient Distributed Property Graph. At a high-level, GraphX extends the Spark RDD abstraction by introducing the Resilient Distributed Property Graph (a directed multigraph with properties attached to each vertex and edge).
5. **MLlib (MachineLearning)**  
   MLlib stands for Machine Learning Library. Spark MLlib is used to perform machine learning in Apache Spark.
6. ***SparkR***It is an R package that provides a distributed data frame implementation. It also supports operations like selection, filtering, aggregation but on large data-sets.

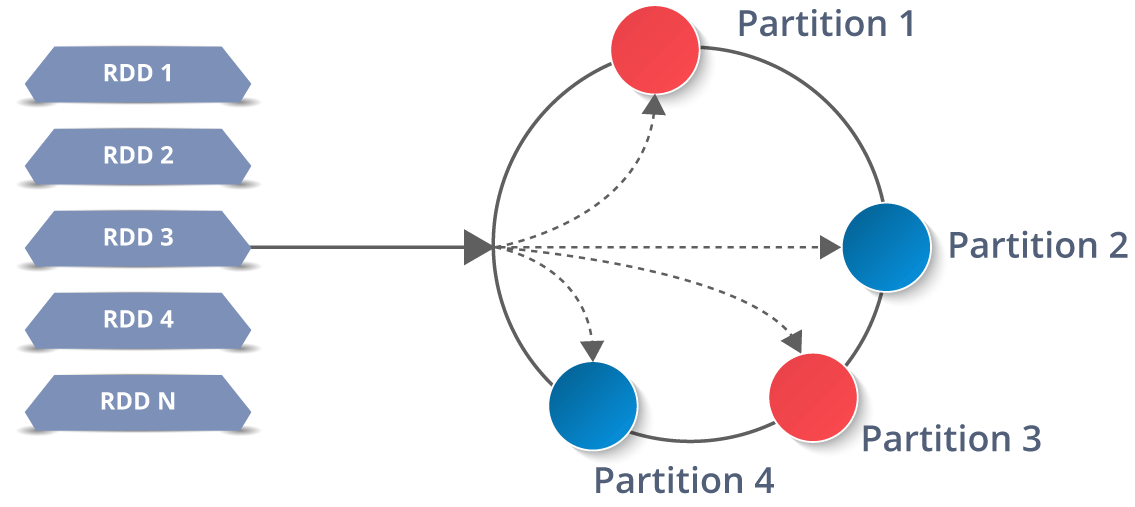
As you can see, Spark comes packed with high-level libraries, including support for R, SQL, Python, Scala, Java etc. These standard libraries increase the seamless integrations in a complex workflow. Over this, it also allows various sets of services to integrate with it like MLlib, GraphX, SQL + Data Frames, Streaming services etc. to increase its capabilities.

Now, let’s discuss the fundamental Data Structure of Spark, i.e. RDD.

## ****Resilient Distributed Dataset(RDD)****

RDDs are the building blocks of any Spark application. RDDs Stands for:

* **Resilient:** Fault tolerant and is capable of rebuilding data on failure
* ***Distributed:*** Distributed data among the multiple nodes in a cluster
* **Dataset:** Collection of partitioned data with values

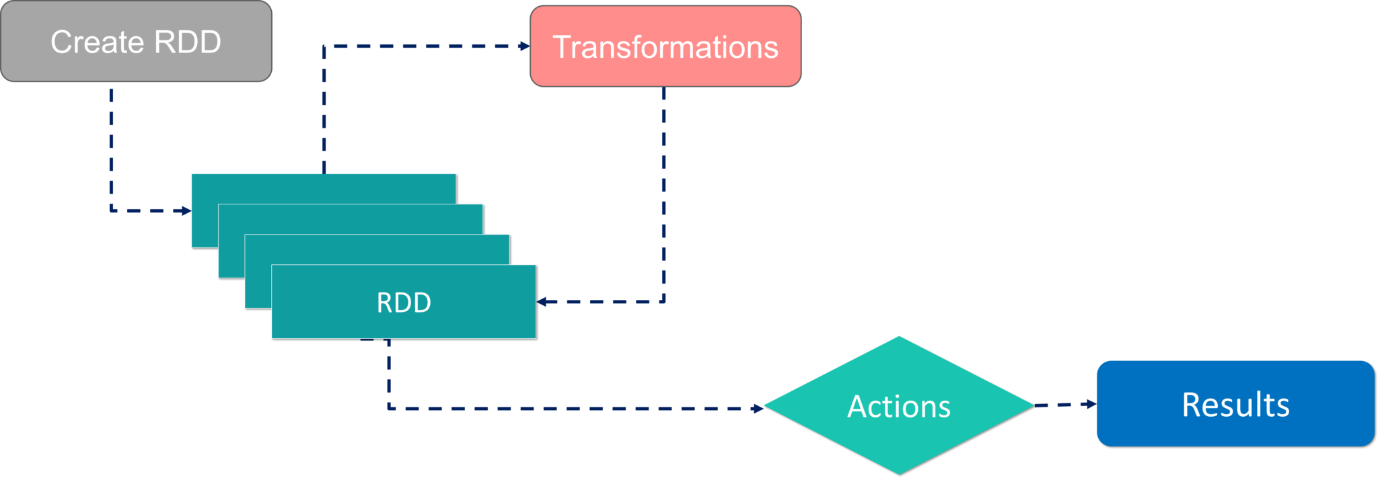


It is a layer of abstracted data over the distributed collection. It is immutable in nature and follows [*lazy transformations*](https://www.edureka.co/blog/spark-tutorial/#Spark_Features).

Now you might be wondering about its working. Well, the data in an RDD is split into chunks based on a key. RDDs are highly resilient, i.e, they are able to recover quickly from any issues as the same data chunks are replicated across multiple executor nodes. Thus, even if one executor node fails, another will still process the data. This allows you to perform your functional calculations against your dataset very quickly by harnessing the power of multiple nodes.

Moreover, once you create an RDD it becomes **immutable**. By immutable I mean, an object whose state cannot be modified after it is created, but they can surely be transformed.

Talking about the distributed environment, each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. Due to this, you can perform transformations or actions on the complete data parallelly. Also, you don’t have to worry about the distribution, because Spark takes care of that.

                                                                   Workflow of RDD

There are two ways to create RDDs − parallelizing an existing collection in your driver program, or by referencing a dataset in an external storage system, such as a shared file system, HDFS, HBase, etc.

With RDDs, you can perform two types of operations:

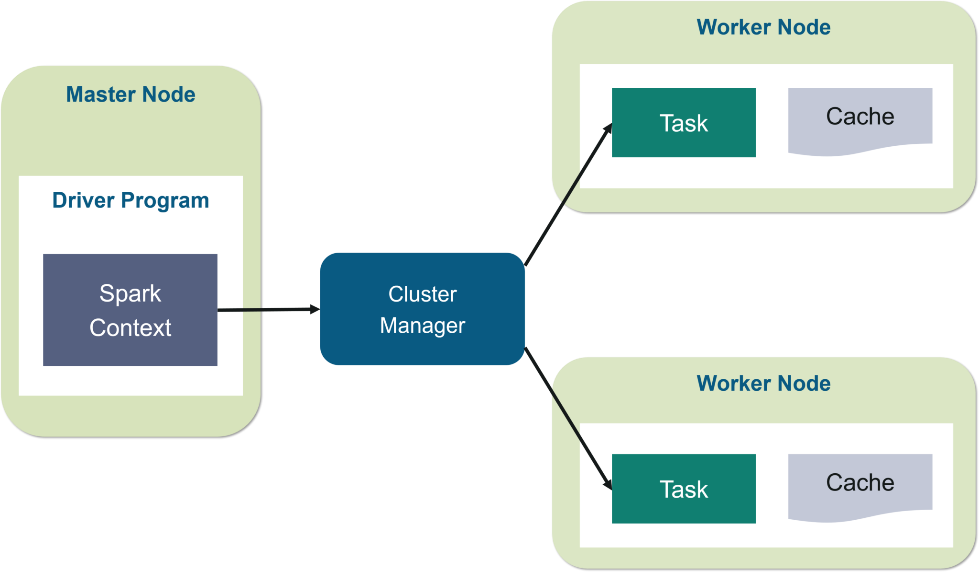
1. **Transformations:**They are the operations that are applied to create a new RDD.
2. **Actions:** They are applied on an RDD to instruct Apache Spark to apply computation and pass the result back to the driver.

I hope you got a thorough understanding of RDD concepts. Now let’s move further and see the working of Spark Architecture.

## ****Working of Spark Architecture****

As you have already seen the basic architectural overview of Apache Spark, now let’s dive deeper into its working.

In your **master node**, you have the driver program, which drives your application. The code you are writing behaves as a driver program or if you are using the interactive shell, the shell acts as the driver program.

                                  Fig: Spark Architecture

Inside the driver program, the first thing you do is, you create a ***Spark Context.*** Assume that the Spark context is a gateway to all the Spark functionalities. It is similar to your database connection. Any command you execute in your database goes through the database connection. Likewise, anything you do on Spark goes through Spark context.

Now, this Spark context works with the **cluster manager** to manage various jobs. The driver program & Spark context takes care of the job execution within the cluster. A job is split into multiple tasks which are distributed over the worker node. Anytime an RDD is created in Spark context, it can be distributed across various nodes and can be cached there.

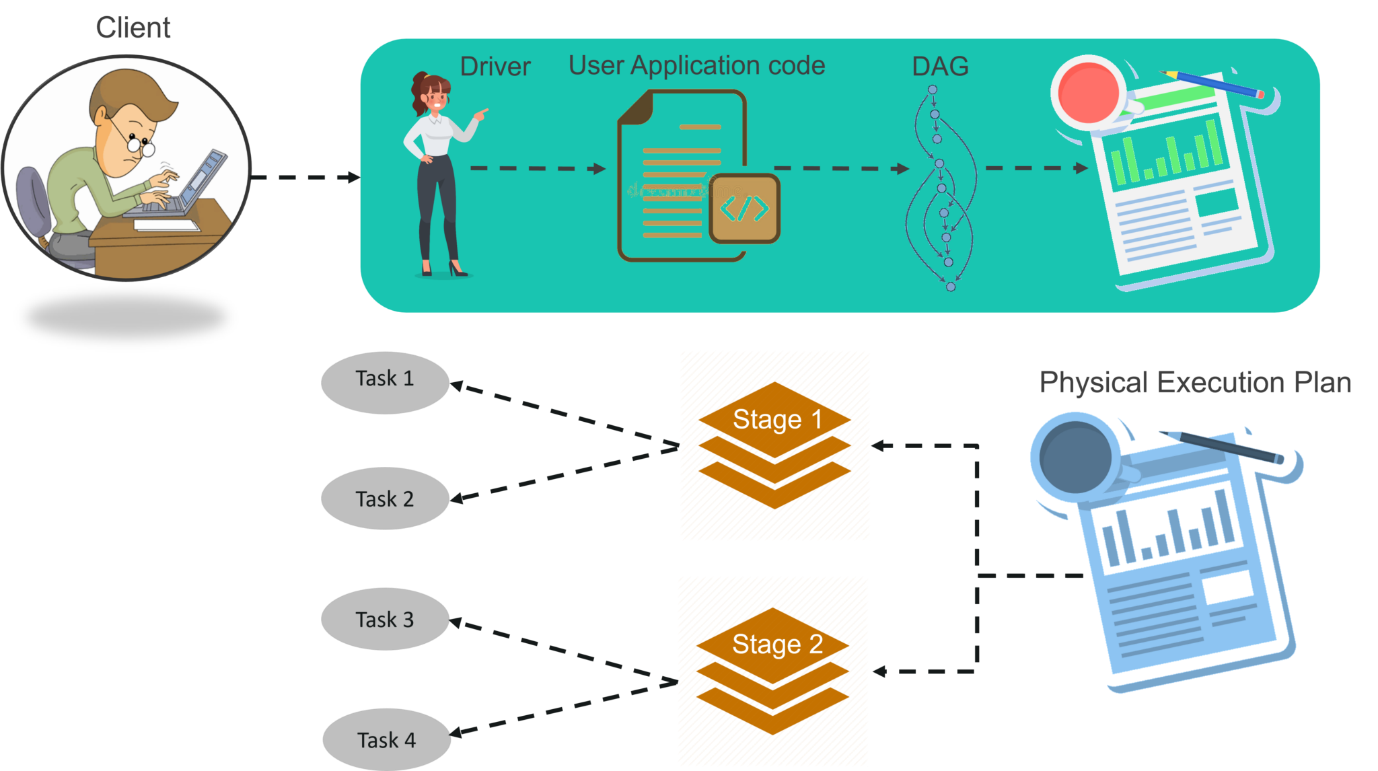
**Worker nodes** are the slave nodes whose job is to basically execute the tasks. These tasks are then executed on the partitioned RDDs in the worker node and hence returns back the result to the Spark Context.

Spark Context takes the job, breaks the job in tasks and distribute them to the worker nodes. These tasks work on the partitioned RDD, perform operations, collect the results and return to the main Spark Context.

If you increase the number of workers, then you can divide jobs into more partitions and execute them parallelly over multiple systems. It will be a lot faster.

With the increase in the number of workers, memory size will also increase & you can cache the jobs to execute it faster.

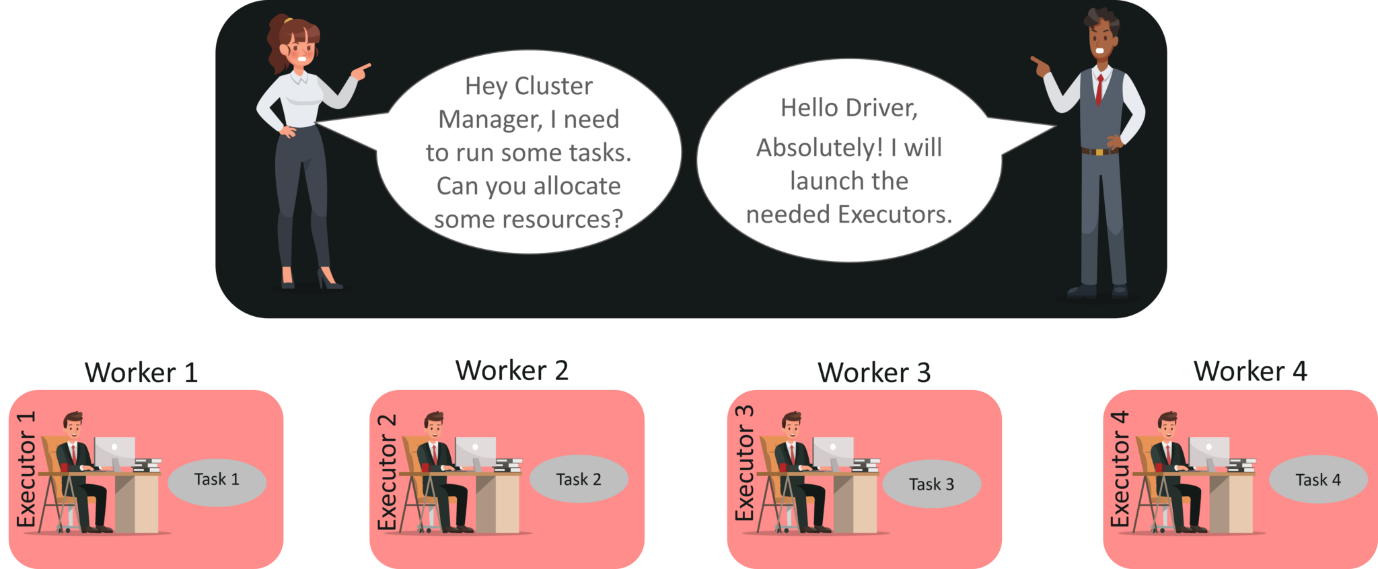
To know about the workflow of Spark Architecture, you can have a look at the **infographic** below:

                                                         Fig: Spark Architecture Infographic

**STEP 1:**The client submits spark user application code. When an application code is submitted, the driver implicitly converts user code that contains transformations and actions into a logically directed acyclic graph called ***DAG.***At this stage, it also performs optimizations such as pipelining transformations.

**STEP 2:** After that, it converts the logical graph called DAG into physical execution plan with many stages. After converting into a physical execution plan, it creates physical execution units called tasks under each stage. Then the tasks are bundled and sent to the cluster.

**STEP 3:** Now the driver talks to the cluster manager and negotiates the resources. Cluster manager launches executors in worker nodes on behalf of the driver. At this point, the driver will send the tasks to the executors based on data placement. When executors start, they register themselves with drivers. So, the driver will have a complete view of executors that are executing the task.

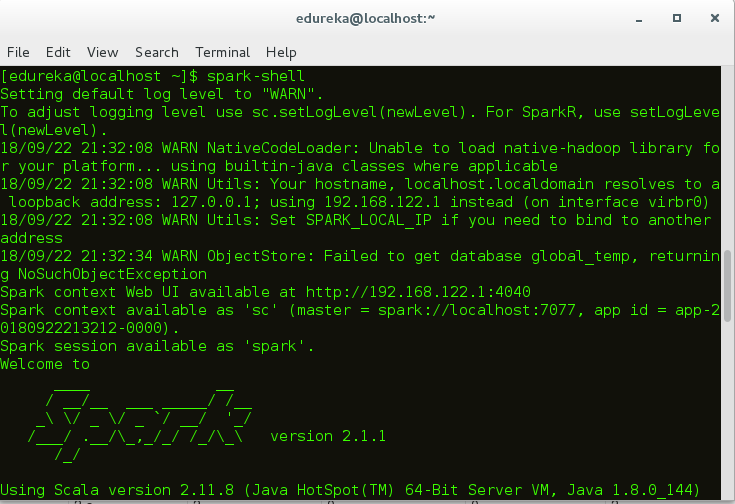


**STEP 4:** During the course of execution of tasks, driver program will monitor the set of executors that runs. Driver node also schedules future tasks based on data placement.

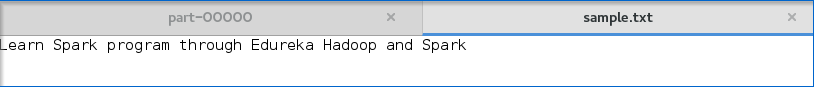
This was all about Spark Architecture. Now, let’s get a hand’s on the working of a Spark shell.

## ****Example using Scala in Spark shell****

At first, let’s start the Spark shell by assuming that Hadoop and Spark daemons are up and running. **Web UI** port for Spark is **localhost:4040.**

                                                                Fig: Spark-shell

Once you have started the Spark shell, now let’s see how to execute a word count example:

1. In this case, I have created a simple text file and stored it in the hdfs directory. You can also use other large data files as well.  
                                                                                     Fig: Input text file
2. Once the spark shell has started, let’s create an RDD. For this, you have to specify the input file path and apply the transformation **flatMap()**. Below code illustrates the same:

|  |  |
| --- | --- |
| 1 | scala> var map = sc.textFile("hdfs://localhost:9000/Example/sample.txt").flatMap(line => line.split(" ")).map(word => (word,1)); |

    3. On executing this code, an RDD will be created as shown in the figure.

                                                                                     Fig: RDD creation

     4.  After that, you need to apply the action **reduceByKey()** to the created RDD.

|  |  |
| --- | --- |
| 1 | scala> var counts = map.reduceByKey(\_+\_); |

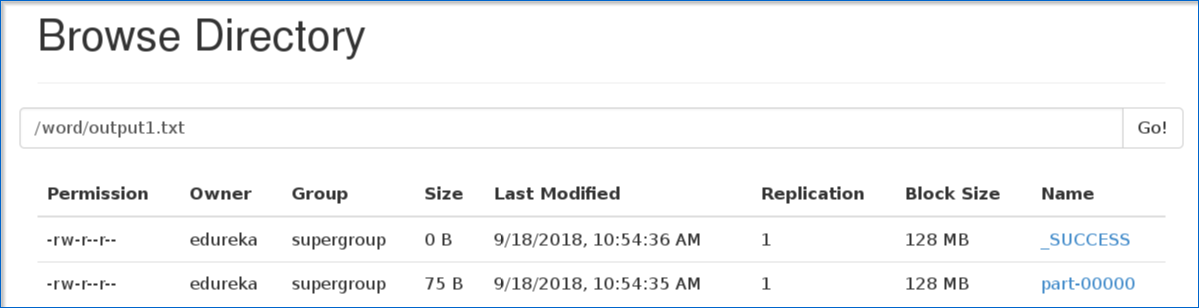
After applying action, execution starts as shown below.

                                                                             Fig: Spark execution in the shell

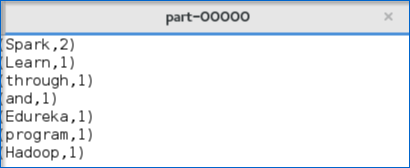
      5.  Next step is to save the output in a text file and specify the path to store the output.

                                                Fig: Specifying the Output path

      6. After specifying the output path, go to the hdfs web browser***localhost:50040.***Here you can see the output text in the ‘part’ file as shown below.

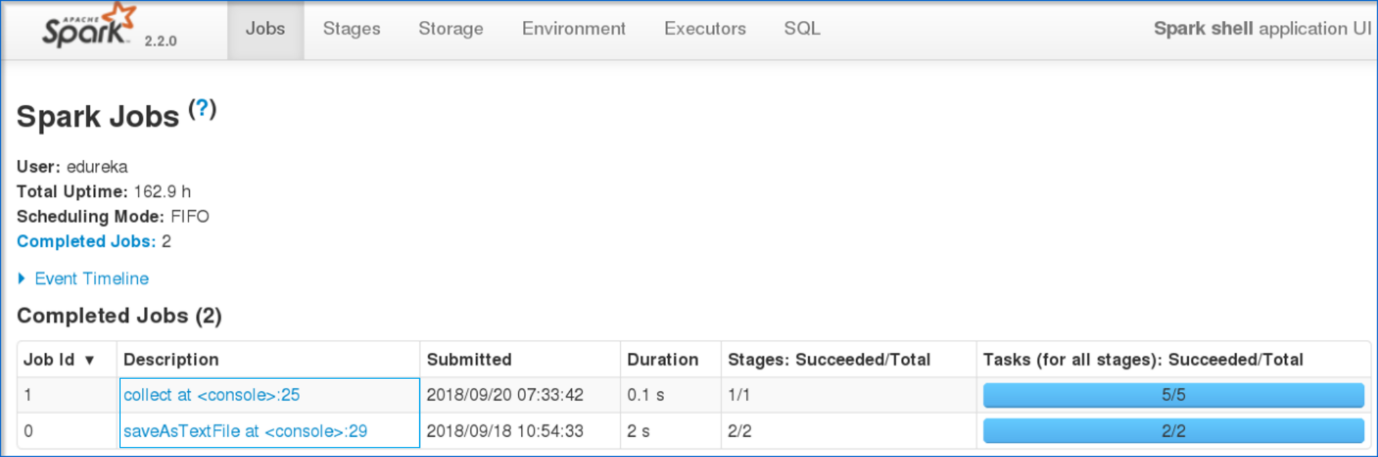
                                                                          Fig: Output part file

       7. Below figure shows the output text present in the ‘part’ file.

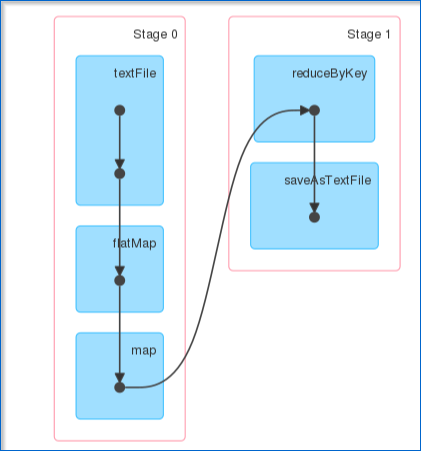
                                  Fig: Output text

I hope that you have understood how to create a Spark Application and arrive at the output.

Now, let me take you through the web UI of Spark to understand the DAG visualizations and partitions of the executed task.

                                                                        Fig: Spark Web User Interface

* On clicking the task that you have submitted, you can view the Directed Acyclic Graph (DAG) of the completed job.

                            Fig: DAG Visualization

* Also, you can view the summary metrics of the executed task like – time taken to execute the task, job ID, completed stages, host IP Address etc.

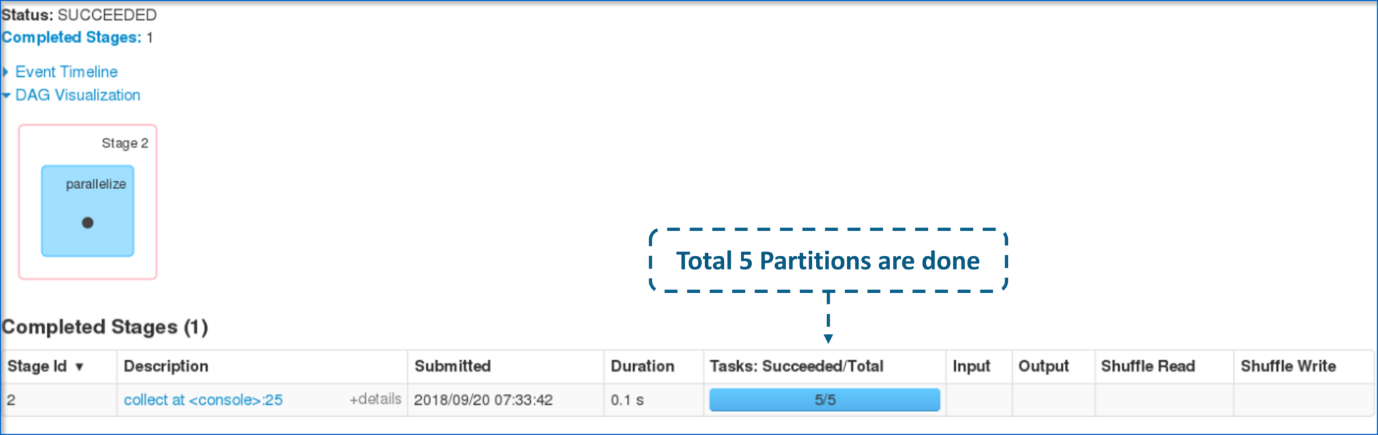
Now, let’s understand about partitions and parallelism in RDDs.

* A ***partition*** is a *logical* *chunk* of a *large* *distributed* *data* *set.*
* By default, Spark tries to *read* *data* *into* *an* *RDD* from the *nodes* that are *close* *to* *it.*

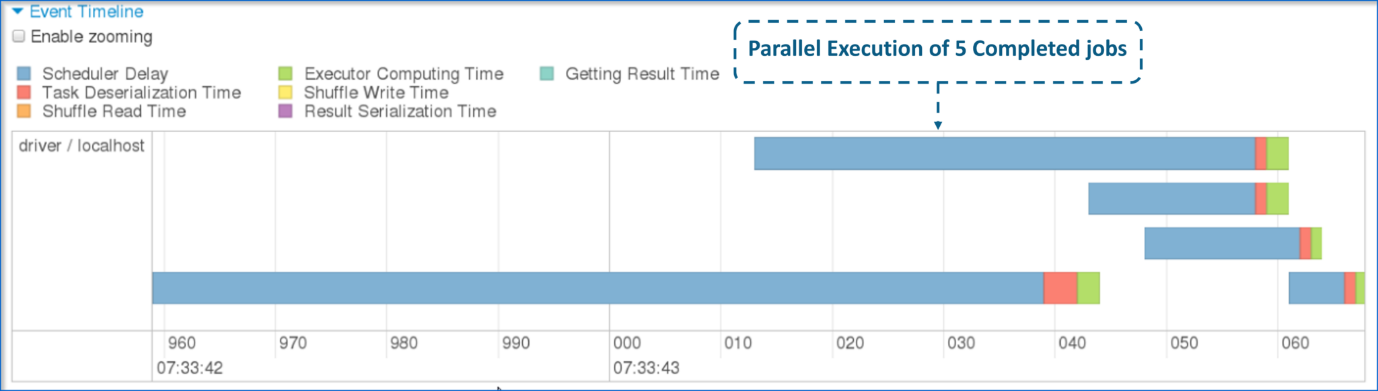
Now, let’s see how to execute a parallel task in the shell.

                                                         Fig: Parallel Execution of a task

* Below figure shows the total number of partitions on the created RDD.

                                                  Fig: Partitions of the completed task

* Now, let me show you how parallel execution of 5 different tasks appears.

                                                              Fig: Parallelism of the 5 completed tasks