Spark Tutorial – Learn Spark Programming

1. Objective – Spark Tutorial

In this Spark Tutorial, we will see an overview of Spark in Big Data. We will start with an introduction to Apache Spark Programming. Then we will move to know the Spark History. Moreover, we will learn why Spark is needed. Afterward, will cover all fundamental of Spark components. Furthermore, we will learn about Spark’s core abstraction and Spark RDD. For more detailed insights, we will also cover spark features, Spark limitations, and Spark Use cases.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/03/Spark-Tutorial-01.jpg)

*Apache Spark Tutorial – What is Apache Spark?*

2. Introduction to Spark Programming

What is Spark? *Spark Programming is nothing but a general-purpose & lightning fast cluster computing platform*. In other words, *it is an open source, wide range data processing engine*. That reveals development API’s, which also qualifies data workers to accomplish streaming, machine learning or SQL workloads which demand repeated access to data sets. However, Spark can perform [**batch processing and stream processing**](https://data-flair.training/blogs/batch-processing-vs-real-time-processing/). Batch processing refers, to the processing of the previously collected job in a single batch. Whereas stream processing means to deal with Spark streaming data.

Moreover, it is designed in such a way that it integrates with all the[**Big data**](https://data-flair.training/blogs/what-is-big-data/) tools. Like spark can access any [**Hadoop**](https://data-flair.training/blogs/hadoop-tutorial-for-beginners/) data source, also can run on Hadoop clusters. Furthermore, Apache Spark extends Hadoop MapReduce to the next level. That also includes iterative queries and stream processing.  
One more common belief about Spark is that it is an extension of Hadoop. Although that is not true. However, Spark is independent of Hadoop since it has its own [**cluster management**](https://data-flair.training/blogs/apache-spark-cluster-managers-tutorial/) system. Basically, it uses Hadoop for storage purpose only.

Although, there is one spark’s key feature that it has in-memory cluster computation capability. Also increases the processing speed of an application.

Basically, Apache Spark offers high-level APIs to users, such as [**Java**](https://data-flair.training/blogs/java-tutorial/), Scala,[**Python**](https://data-flair.training/blogs/python-tutorial-for-beginners/)**,**and R. Although, Spark is written in Scala still offers rich APIs in Scala, Java, Python, as well as R. We can say, it is a tool for running spark applications.

Most importantly, by [**comparing Spark with Hadoop**](https://data-flair.training/blogs/apache-spark-vs-hadoop-mapreduce/)**,** it is 100 times faster than Hadoop In-Memory mode and 10 times faster than Hadoop  On-Disk mode.

3. Spark Tutorial –  History

At first, in 2009 Apache Spark was introduced in the UC Berkeley R&D Lab, which is now known as AMPLab. Afterward, in 2010 it became open source under BSD license. Further, the spark was donated to Apache Software Foundation, in 2013. Then in 2014, it became top-level Apache project.

4. Why Spark?

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/03/Why-Spark-01-1.jpg)

*Spark Tutorial – Why Spark?*

As we know, there was no general purpose computing engine in the industry, since

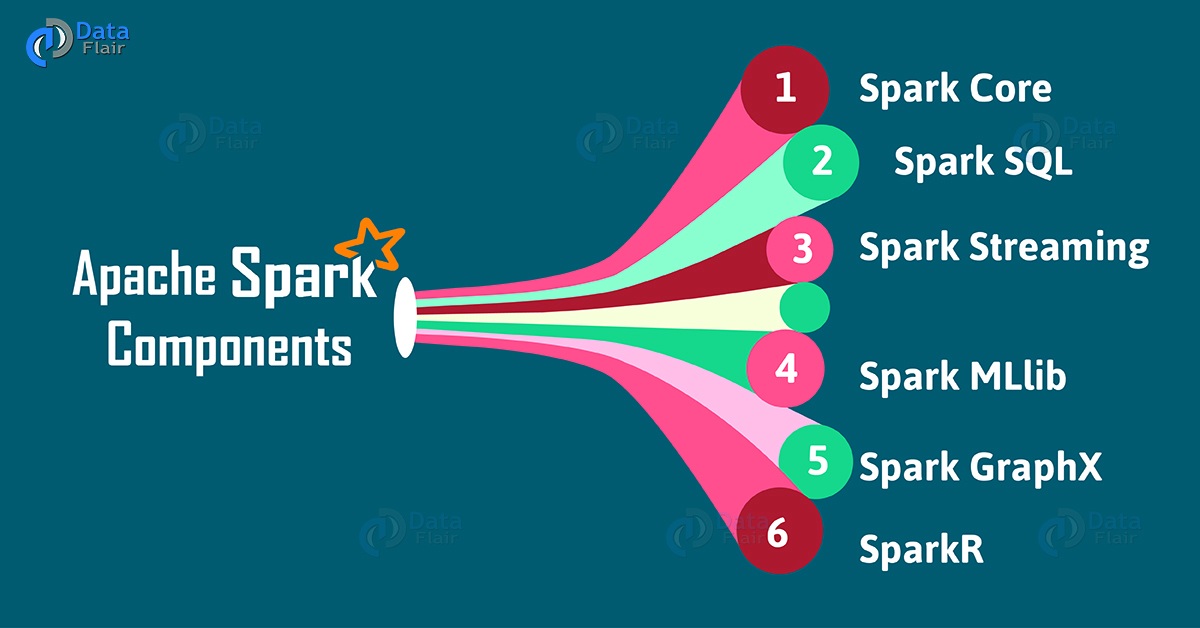
1. To perform batch processing, we were using [**Hadoop MapReduce**](https://data-flair.training/blogs/hadoop-mapreduce-tutorial/).
2. Also, to perform stream processing, we were using Apache Storm / S4.
3. Moreover, for interactive processing, we were using Apache Impala / Apache Tez.
4. To perform graph processing, we were using Neo4j / Apache Giraph.

Hence there was no powerful engine in the industry, that can process the data both in real-time and batch mode. Also, there was a requirement that one engine can respond in sub-second and perform in-memory processing.  
Therefore, Apache Spark programming enters, it is a powerful open source engine. Since, it offers real-time stream processing, interactive processing, graph processing, in-memory processing as well as batch processing. Even with very fast speed, ease of use and standard interface. Basically, these features create the difference between Hadoop and Spark. Also makes a huge [**comparison between Spark vs Storm.**](https://data-flair.training/blogs/apache-storm-vs-spark-streaming/)

5. Apache Spark Components

In this Apache Spark Tutorial, we discuss Spark Components. It puts the promise for faster data processing as well as easier development. It is only possible because of its components. All these Spark components resolved the issues that occurred while using Hadoop MapReduce.

Now let’s discuss each Spark Ecosystem Component one by one-

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/02/Apache-Spark-Components.jpg)

*Spark Tutorial – Apache Spark Ecosystem Components*

a. Spark Core

Spark Core is a central point of Spark. Basically, it provides an execution platform for all the Spark applications. Moreover, to support a wide array of applications, Spark Provides a  generalized platform.

b. Spark SQL

On the top of Spark, Spark**SQL** enables users to run SQL/HQL queries. We can process structured as well as semi-structured data, by using Spark SQL. Moreover, it offers to run unmodified queries up to 100 times faster on existing deployments. To learn[**Spark SQL in detail**](https://data-flair.training/blogs/spark-sql-tutorial/), follow this link.

c. Spark Streaming

Basically, across live streaming, Spark Streaming enables a powerful interactive and data analytics application. Moreover, the live streams are converted into micro-batches those are executed on top of spark core. Learn [**Spark Streaming in detail.**](https://data-flair.training/blogs/apache-spark-streaming-tutorial/)

d. Spark MLlib

Machine learning library delivers both efficiencies as well as the high-quality algorithms. Moreover, it is the hottest choice for a data scientist. Since it is capable of in-memory data processing, that improves the performance of iterative algorithm drastically.

e. Spark GraphX

Basically, Spark GraphX is the graph computation engine built on top of Apache Spark that enables to process graph data at scale.

1.2M

86

machine learning project spam detector

f. SparkR

Basically, to use Apache Spark from [**R**](https://data-flair.training/blogs/r-programming-tutorial/). It is [**R package**](https://data-flair.training/blogs/r-packages-tutorial/) that gives light-weight frontend. Moreover, it allows data scientists to analyze large datasets. Also allows running jobs interactively on them from the R shell. Although, the main idea behind SparkR was to explore different techniques to integrate the usability of R with the scalability of Spark. Follow the link to learn **[SparkR in detail.](https://data-flair.training/blogs/sparkr/)**

To learn about all the components of Spark in detail, follow link [**Apache Spark Ecosystem – Complete Spark Components Guide**](https://data-flair.training/blogs/apache-spark-ecosystem-components/)

6. Resilient Distributed Dataset – RDD

The key abstraction of Spark is RDD. RDD is an acronym for Resilient Distributed Dataset. It is the fundamental unit of data in Spark. Basically, it is a distributed collection of elements across cluster nodes. Also performs parallel operations. Moreover, Spark RDDs are immutable in nature. Although, it can generate new RDD by transforming existing Spark RDD.Learn about [**Spark RDDs in detail.**](https://data-flair.training/blogs/apache-spark-rdd-tutorial/)

a. Ways to create Spark RDD

Basically, there are 3 ways to create Spark RDDs

**i. Parallelized collections**

By invoking parallelize method in the driver program, we can create parallelized collections.

**ii. External datasets**

One can create Spark RDDs, by calling a textFile method. Hence, this method takes URL of the file and reads it as a collection of lines.

**iii. Existing RDDs**

Moreover, we can create new RDD in spark, by applying transformation operation on existing RDDs.

To learn all three [**ways to create RDD**](https://data-flair.training/blogs/create-rdds-in-apache-spark/) in detail, follow the link.

b. Spark RDDs operations

There are two types of operations, which Spark RDDs supports:

**i. Transformation Operations**

It creates a new Spark RDD from the existing one. Moreover, it passes the dataset to the function and returns new dataset.

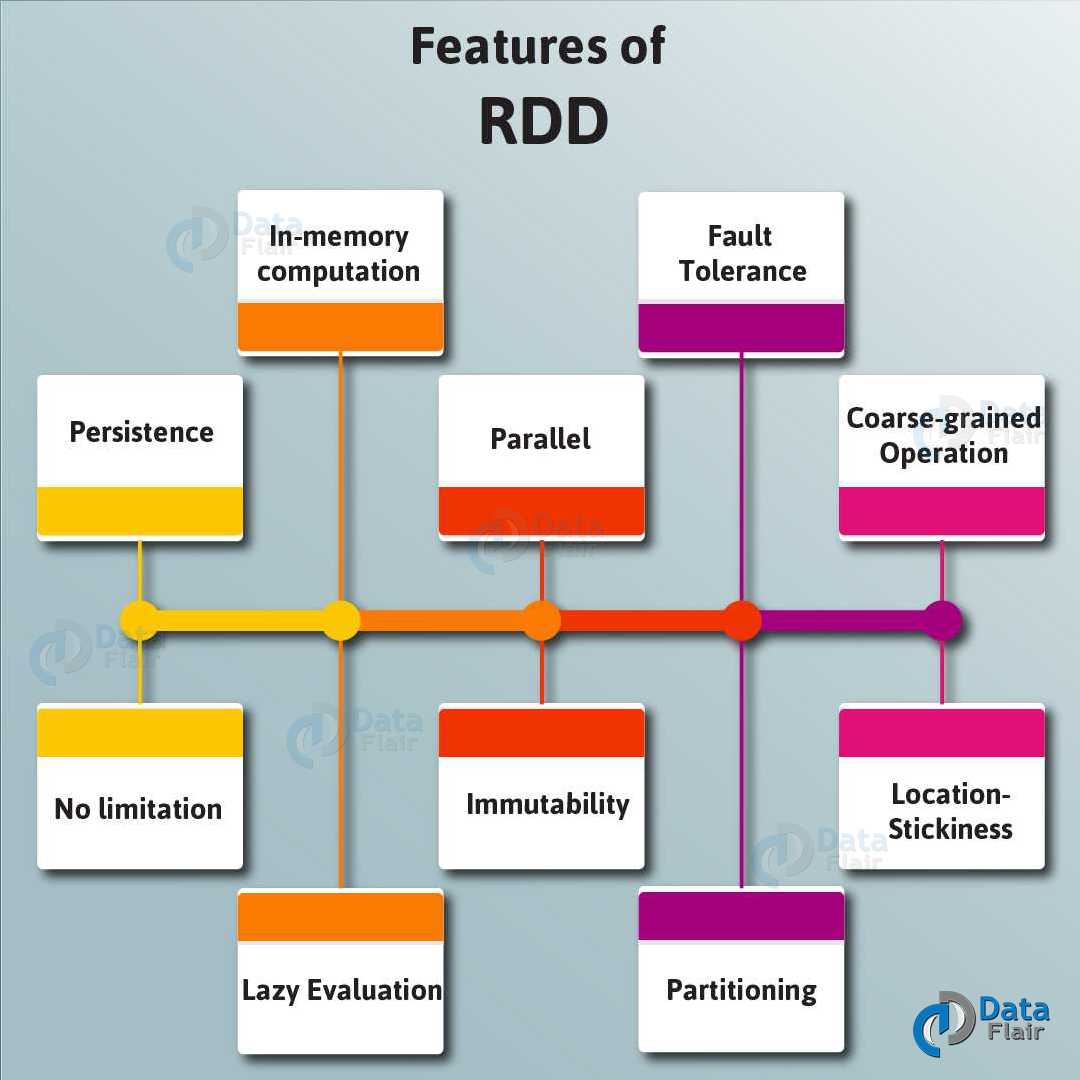
**ii. Action Operations**

In Apache Spark, Action returns final result to driver program or write it to the external data store.

Learn [**RDD Operations in detail.**](https://data-flair.training/blogs/spark-rdd-operations-transformations-actions/)

c. Sparkling Features of Spark RDD

There are various advantages of using RDD. Some of them are

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/02/Features-of-RDD-01-1.jpg)

*Spark Tutorial – Spark RDD Features*

**i. In-memory computation**

Basically, while storing data in RDD, data is stored in memory for as long as you want to store. It improves the performance by an order of magnitudes by keeping the data in memory.

**ii. Lazy Evaluation**

Spark Lazy Evaluation means the data inside RDDs are not evaluated on the go. Basically, only after an action triggers all the changes or the computation is performed. Therefore, it limits how much work it has to do. learn [**Lazy Evaluation in detail.**](https://data-flair.training/blogs/apache-spark-lazy-evaluation/)

**iii. Fault Tolerance**

If any worker node fails, by using lineage of operations, we can re-compute the lost partition of RDD from the original one. Hence, it is possible to recover lost data easily. Learn [**Fault Tolerance in detail**](https://data-flair.training/blogs/fault-tolerance-in-apache-spark/).

**iv. Immutability**

Immutability means once we create an RDD, we can not manipulate it. Moreover, we can create a new RDD by performing any transformation. Also, we achieve consistency through immutability.

**v. Persistence**

In in-memory, we can store the frequently used RDD. Also, we can retrieve them directly from memory without going to disk. It results in the speed of the execution. Moreover, we can perform multiple operations on the same data. It is only possible by storing the data explicitly in memory by calling persist() or cache() function.

Learn[**Persistence and Caching Mechanism**](https://data-flair.training/blogs/apache-spark-rdd-persistence-caching/)in detail.

**vi. Partitioning**

Basically, RDD partition the records logically. Also, distributes the data across various nodes in the cluster. Moreover, the logical divisions are only for processing and internally it has no division. Hence, it provides parallelism.

**vii. Parallel**

While we talk about parallel processing, RDD processes the data parallelly over the cluster.

**viii. Location-Stickiness**

To compute partitions, RDDs are capable of defining placement preference. Moreover, placement preference refers to information about the location of RDD. Although, the DAGScheduler places the partitions in such a way that task is close to data as much as possible. Moreover, it speeds up computation.

**ix. Coarse-grained Operation**

Generally, we apply coarse-grained transformations to Spark RDD. It means the operation applies to the whole dataset not on the single element in the data set of RDD in Spark.

**x. Typed**

There are several types of Spark RDD. Such as: RDD [int], RDD [long], RDD [string].

**xi. No limitation**

There are no limitations to use the number of Spark RDD. We can use any no. of RDDs. Basically, the limit depends on the size of disk and memory.

In this Apache Spark tutorial, we cover most Features of Spark RDD to learn more about [**RDD Features**](https://data-flair.training/blogs/apache-spark-rdd-features/) follow this link.

7. Spark Tutorial – Spark Streaming

While data is arriving continuously in an unbounded sequence is what we call a data stream. Basically, for further processing, Streaming divides continuous flowing input data into discrete units. Moreover, we can say it is a low latency processing and analyzing of streaming data.

In addition, an extension of the core Spark API Streaming was added to Apache Spark in 2013. That offers scalable, fault-tolerant and high-throughput processing of live data streams. Although, here we can do data ingestion from many sources. Such as Kafka, [**Apache Flume**](https://data-flair.training/blogs/apache-flume-tutorial/), Amazon Kinesis or TCP sockets. However, we do processing here by using complex algorithms which are expressed with high-level functions such as map, reduce, join and window.

a. Internal working of Spark Streaming

Let’s understand its internal working. While live input data streams are received. It further divided into batches by Spark streaming, Afterwards, these batches are processed by the Spark engine to generate the final stream of results in batches.

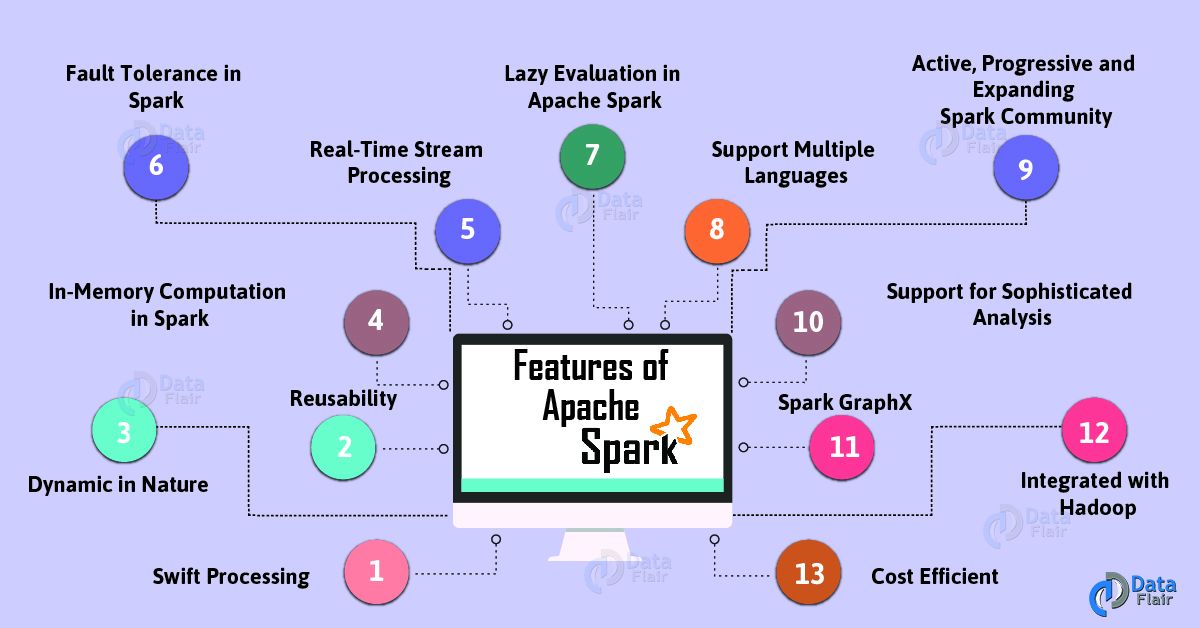
b. Discretized Stream (DStream)

Apache Spark Discretized Stream is the key abstraction of Spark Streaming. That is what we call Spark DStream. Basically, it represents a stream of data divided into small batches. Moreover, DStreams are built on Spark RDDs, Spark’s core data abstraction. It also allows Streaming to seamlessly integrate with any other Apache Spark components. Such as Spark MLlib and Spark SQL.

Follow this link, to Learn [**Concept of Dstream in detail.**](https://data-flair.training/blogs/apache-spark-dstream-discretized-streams/)

8. Features of Apache Spark

There are several sparkling Apache Spark features:

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/01/Features-of-Apache-Spark-01.jpg)

*Apache Spark Tutorial – Features of Apache Spark*

**a. Swift Processing**

Apache Spark offers high data processing speed. That is about 100x faster in memory and 10x faster on the disk. However, it is only possible by reducing the number of read-write to disk.

**b. Dynamic in Nature**

Basically, it is possible to develop a parallel application in Spark. Since there are 80 high-level operators available in Apache Spark.

**c. In-Memory Computation in Spark**

The increase in processing speed is possible due to [**in-memory processing**](https://data-flair.training/blogs/apache-spark-in-memory-computing/). It enhances the processing speed.

**d. Reusability**

We can easily reuse spark code for batch-processing or join stream against historical data. Also to run ad-hoc queries on stream state.

**e. Spark Fault Tolerance**

Spark offers fault tolerance. It is possible through Spark’s core abstraction-RDD. Basically, to handle the failure of any worker node in the cluster, Spark RDDs are designed. Therefore, the loss of data is reduced to zero.

**f. Real-Time Stream Processing**

We can do real-time stream processing in Spark. Basically, Hadoop does not support real-time processing. It can only process data which is already present. Hence with Spark Streaming, we can solve this problem.

**g. Lazy Evaluation in Spark**

All the transformations we make in Spark RDD are Lazy in nature, that is it does not give the result right away rather a new RDD is formed from the existing one. Thus, this increases the efficiency of the system.

**h. Support Multiple Languages**

Spark supports multiple languages. Such as Java, R, [**Scala**](https://data-flair.training/blogs/why-you-should-learn-scala-introductory-tutorial/), Python. Hence, it shows dynamicity. Moreover, it also overcomes the [**limitations of Hadoop**](https://data-flair.training/blogs/limitations-of-hadoop/) since it can only build applications in [**Java.**](https://data-flair.training/blogs/features-of-java/)

**i. Support for Sophisticated Analysis**

There are dedicated tools in Apache Spark. Such as for streaming data interactive/declarative queries, machine learning which add-on to map and reduce.

**j. Integrated with Hadoop**

As we know Spark is flexible. It can run independently and also on Hadoop YARN Cluster Manager. Even it can read existing Hadoop data.

**k. Spark GraphX**

In Spark, a component for graph and graph-parallel computation, we have GraphX. Basically, it simplifies the graph analytics tasks by the collection of graph algorithm and builders.

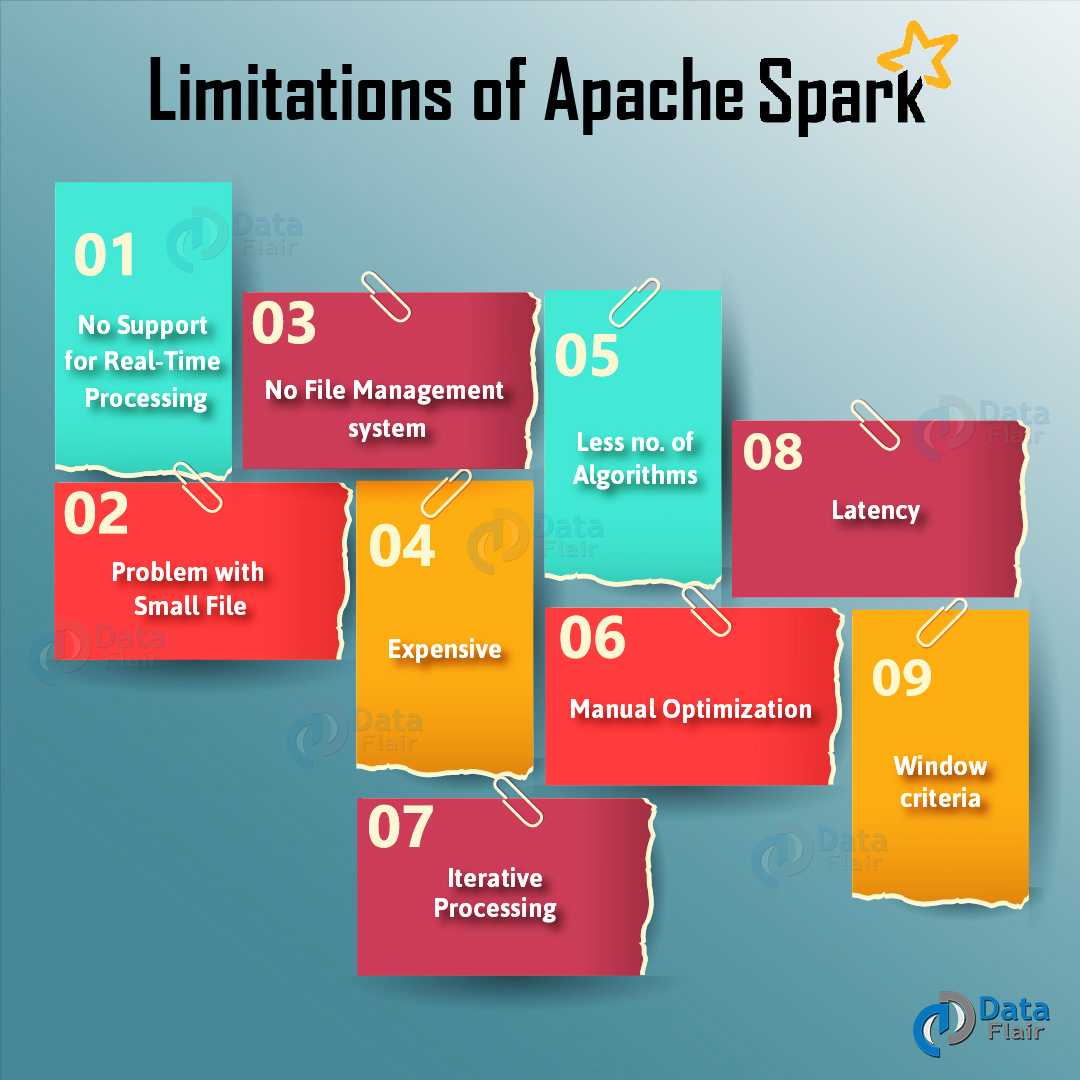
**l. Cost Efficient**

For Big data problem as in Hadoop, a large amount of storage and the large data center is required during replication. Hence, Spark programming turns out to be a cost-effective solution.

Learn[**All features of Apache Spark, in detail.**](https://data-flair.training/blogs/apache-spark-features/)

9. Limitations of Apache Spark Programming

There are many limitations of Apache Spark. Let’s learn all one by one:

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/02/Limitations-of-Apache-Spark-01.jpg)

*Spark Tutorial – Limitations of Apache Spark Programming*

**a. No Support for Real-time Processing**

Basically, Spark is near real-time processing of live data. In other words, Micro-batch processing takes place in Spark Streaming. Hence we can not say Spark is completely Real-time Processing engine.

**b. Problem with Small File**

In RDD, each file is a small partition. It means, there is the large amount of tiny partition within an RDD. Hence, if we want efficiency in our processing, the RDDs should be repartitioned into some manageable format. Basically, that demands extensive shuffling over the network.

**c. No File Management System**

A major issue is Spark does not have its own file management system. Basically, it relies on some other platform like Hadoop or another cloud-based platform.

**d. Expensive**

While we desire cost-efficient processing of big data, Spark turns out to be very expensive. Since keeping data in memory is quite expensive. However the memory consumption is very high, and it is not handled in a user-friendly manner. Moreover, we require lots of RAM to run in-memory, thus the cost of spark is much higher.

**e. Less number of Algorithms**

[**Spark MLlib**](https://data-flair.training/blogs/apache-spark-mllib/) have very less number of available algorithms. For example, Tanimoto distance.

**f. Manual Optimization**

It is must that Spark job is manually optimized and is adequate to specific datasets. Moreover, to partition and cache in spark to be correct, it is must to control it manually.

**g. Iterative Processing**

Basically, here data iterates in batches. Also, each iteration is scheduled and executed separately.

**h. Latency**

On comparing with **[Flink](https://data-flair.training/blogs/apache-flink-tutorial-comprehensive-guide/)**, Apache Spark has higher latency.

**i. Window Criteria**

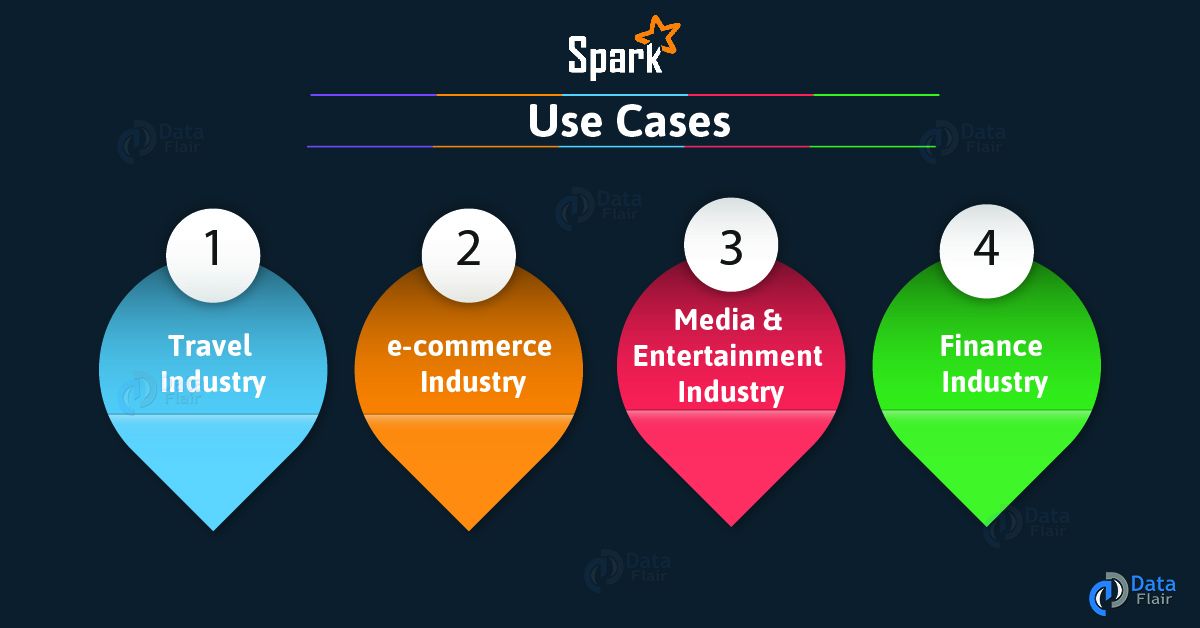
Spark only support time-based window criteria not record based window criteria.

**Note**: To overcome these limitations of Spark, we can use [**Apache Flink – 4G of Big Data**](https://data-flair.training/blogs/real-life-apache-flink-use-cases/).

[**Learn All Limitations of Apache Spark, in detail.**](https://data-flair.training/blogs/limitations-of-apache-spark/)

10. Apache Spark Tutorial – Use Cases

There are many industry-specific Apache Spark use cases, let’s discuss them one by one:

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/01/Spark-Use-Cases-01-1.jpg)

*Spark Tutorial – Apache Spark Use Cases*

**a. Spark Use Cases in the Finance Industry**

There are many banks those are using Spark. Basically, it helps to access and analyze many of the parameters in the bank sector like the emails, social media profiles, call recordings, forum, and many more. Further, it helps to make right decisions for several zones.

**b. Apache Spark Use Cases in E-Commerce Industry**

Basically, it helps with information about a real-time transaction. Moreover, those are passed to streaming clustering algorithms.

**c. Apache Spark Use Cases in Media & Entertainment Industry**

We use Spark to identify patterns from the real-time in-game events. Moreover, it helps to respond in order to harvest lucrative business opportunities.

**d. Apache Spark Use Cases in Travel Industry**

Basically, travel industries are using spark rapidly. Moreover, it helps users to plan a perfect trip by speed up the personalized recommendations. Although, its review process of the hotels in a readable format is done by using Spark.

Apache Spark tutorial cover Spark real-time use Cases, there are many more, follow the link to learn all in detail. [**Apache Spark use cases in real time**](https://data-flair.training/blogs/apache-spark-use-cases-in-real-time/)

11. Spark Tutorial – Conclusion

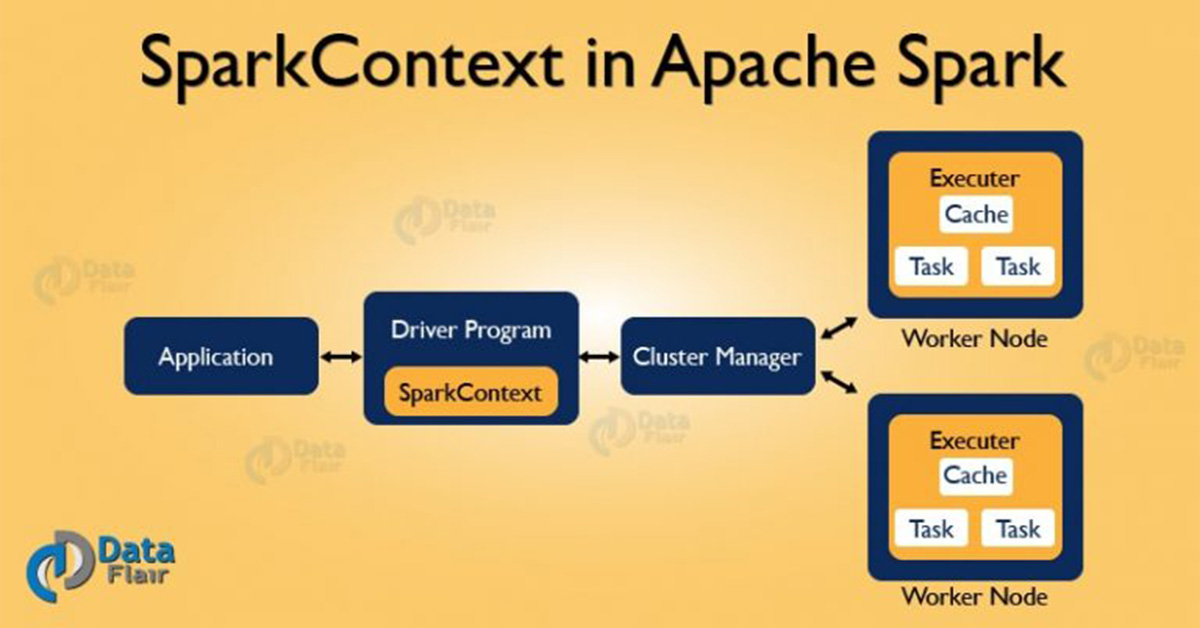
As a result, we have seen every aspect of Apache Spark, what is Apache spark programming and spark definition, History of Spark, why Spark is needed, Components of Apache Spark, Spark RDD, Features of Spark RDD, Spark Streaming, Features of Apache Spark, Limitations of Apache Spark, Apache Spark use cases. In this tutorial we were trying to cover all spark notes, hope you get desired information in it if you feel to ask any query, feel free to ask in the comment section.

Learn SparkContext – Introduction and Functions

1. Objective

**SparkContext** is the entry gate of [**Apache Spark**](http://data-flair.training/blogs/apache-spark-tutorial-quickstart-introduction/) functionality. The most important step of any Spark driver application is to generate SparkContext. It allows your Spark Application to access Spark Cluster with the help of Resource Manager (**YARN/Mesos**). To create SparkContext, first **SparkConf** should be made. The SparkConf has a configuration parameter that our Spark driver application will pass to SparkContext.  
In this Apache Spark tutorial, we will deeply understand what is SparkContext in Spark. How to create SparkContext Class in Spark with the help of Spark-Scala word count program. We will also learn various tasks of SparkContext and how to stop SparkContext in Apache Spark.

So, let’s start SparkContext tutorial.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2017/04/SparkContext-Apache-Spark-768x402-1.jpg)

*Learn SparkContext – Introduction and Functions*

Learn [**how to install Apache Spark in standalone mode**](http://data-flair.training/blogs/apache-spark-installation-in-standalone-mode/)**and**[**Apache Spark installation in a multi-node cluster.**](http://data-flair.training/blogs/apache-spark-installation-on-multi-node-cluster-step-by-step-guide/)

2. What is SparkContext in Apache Spark?

SparkContext is the entry point of Spark functionality. The most important step of any Spark driver application is to generate SparkContext. It allows your Spark Application to access Spark Cluster with the help of Resource Manager. The resource manager can be one of these three- [**Spark Standalone**](http://data-flair.training/blogs/apache-spark-cluster-managers-tutorial/),  [**YARN**](http://data-flair.training/blogs/hadoop-yarn-tutorial/), [**Apache Mesos**.](http://data-flair.training/blogs/apache-mesos-tutorial-learn-mesos/)

3. How to Create SparkContext Class?

If you want to create SparkContext, first **SparkConf** should be made. The SparkConf has a configuration parameter that our Spark driver application will pass to SparkContext. Some of these parameter defines properties of Spark driver application. While some are used by Spark to allocate resources on the cluster, like the number, memory size, and cores used by executor running on the worker nodes.  
In short, it guides how to access the Spark cluster. After the creation of a SparkContext object, we can invoke functions such as **textFile, sequenceFile, parallelize** etc. The different contexts in which it can run are local, yarn-client, Mesos URL and Spark URL.  
Once the SparkContext is created, it can be used to [**create RDDs**](http://data-flair.training/blogs/how-to-create-rdds-in-apache-spark/), broadcast variable, and accumulator, ingress Spark service and run jobs. All these things can be carried out until SparkContext is stopped.

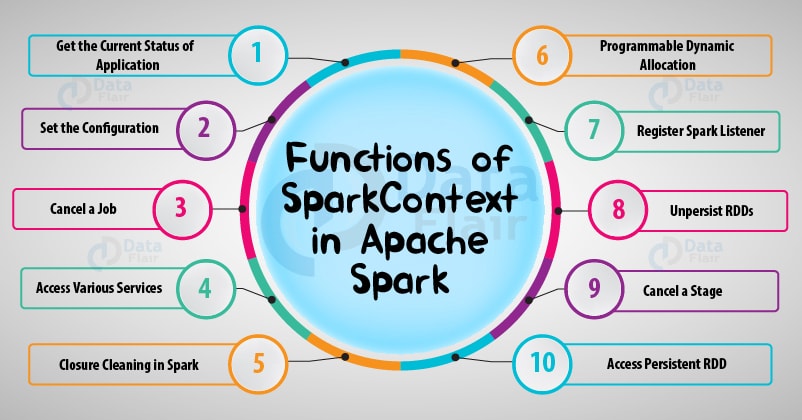
4. Stopping SparkContext

Only one SparkContext may be active per JVM. You must stop the active it before creating a new one as below:  
stop(): Unit  
It will display the following message:  
***INFO SparkContext: Successfully stopped SparkContext***

5. Spark Scala Word Count Example

Let’s see how to create SparkContext using SparkConf with the help of Spark-Scala word count example-  
[php]  
package com.dataflair.spark  
import org.apache.spark.SparkContext  
import org.apache.spark.SparkConf  
object Wordcount {  
def main(args: Array[String]) {  
//Create conf object  
val conf = new SparkConf()  
.setAppName(“WordCount”)  
//create spark context object  
val sc = new SparkContext(conf)  
//Check whether sufficient params are supplied  
if (args.length < 2) {  
println(“Usage: ScalaWordCount <input> <output>”)  
System.exit(1)  
}  
//Read file and create RDD  
val rawData = sc.textFile(args(0))  
//convert the lines into words using flatMap operation  
val words = rawData.flatMap(line => line.split(” “))  
//count the individual words using map and reduceByKey operation  
val wordCount = words.map(word => (word, 1)).reduceByKey(\_ + \_)  
//Save the result  
wordCount.saveAsTextFile(args(1))  
//stop the spark context  
sc.stop  
}  
}[/php]

6. Functions of SparkContext in Apache Spark

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2017/08/functions-of-sparkcontext-in-apache-spark.jpg)

*10 Important Functions of SparkContext in Apache Spark*

i. To get the current status of Spark Application

* **SpkEnv –**It is a runtime environment with Spark’s public services. It interacts with each other to establish a distributed computing platform for Spark Application. A SparkEnv object that holds the required runtime services for[**running Spark application**](http://data-flair.training/blogs/how-apache-spark-works-run-time-spark-architecture/) with the different environment for the driver and executor represents the Spark runtime environment.
* **SparkConf –**The Spark Properties handles maximum applications settings and are configured separately for each application. We can also easily set these properties on a SparkConf. Some common properties like master URL and application name, as well as an arbitrary key-value pair, configured through the**set()** method.
* **Deployment environment (as master URL) –**Spark deployment environment are of two types namely local and clustered. **Local mode** is non-distributed single-JVM deployment mode. All the execution components – driver, executor, LocalSchedulerBackend, and master are present in same single JVM. Hence, the only mode where drivers are useful for execution is the local mode. For testing, debugging or demonstration purpose, the local mode is suitable because it requires no earlier setup to launch spark application. While in clustered mode, the Spark runs in distributive mode. [**Learn Spark Cluster Manager in detail.**](http://data-flair.training/blogs/apache-spark-cluster-managers-tutorial/)

ii. To set the configuration

* **Master URL –**The master method returns back the current value of *spark.master* which is deployment environment in use.
* **Local properties-Creating Logical Job Groups –**The reason of local properties concept is to form logical groups of jobs by means of properties that create the separate job launched from different threads belong to a single logic group.We can set a local property which will affect Spark jobs submitted from a thread, such as the Spark fair scheduler pool.
* **Default Logging level –**It lets you set the root login level in a Spark application, for example, [Spark Shell.](http://data-flair.training/blogs/apache-spark-shell-commands-beginners-tutorial/)

iii. To Access various services

It also helps in accessing services like *TaskScheduler, LiveListenBus, BlockManager, SchedulerBackend, ShuffelManager* and the optional *ContextCleaner*.

iv. To Cancel a job

*cancleJob* simply requests *DAGScheduler* to drop a Spark job.  
Learn about Spark [**DAG(Directed Acyclic Graph)**](http://data-flair.training/blogs/directed-acyclic-graph-dag-in-apache-spark/) in detail.

v. To Cancel a stage

cancleStage simply requests *DAGScheduler* to drop a Spark stage.

vi. For Closure cleaning in Spark

Spark cleanups the closure every time an Action occurs, i.e. the body of Action before it is serialized and sent over the wire to execute. The clean method in SparkContext does this. This, in turn, calls *ClosureClean.clean* method. It not only cleans the closure but also referenced closure is clean transitively. It assumes serializable until it does not explicitly reference unserializable objects.

vii. To Register Spark listener

We can register a custom *SparkListenerInterface* with the help of *addSparkListener* method. We can also register custom listeners using the *spark.extraListeners* setting.

viii. Programmable Dynamic allocation

It also provides the following method as the developer API for dynamic allocation of executors: *requestExecutors, killExecutors, requestTotalExecutors, getExecutorIds*.

ix. To access persistent RDD

*getPersistentRDDs*gives the collection of **RDDs** that have marked themselves as persistent via cache.

x. To unpersist RDDs

From the master’s Block Manager and the internal *persistentRdds*mapping, the unpersist removes the RDD.

So, this was all in Sparkcontext Tutorial. Hope you like our explanation.

## 7. Conclusion

Hence, SparkContext provides the various functions in Spark like get the current status of Spark Application, set the configuration, cancel a job, Cancel a stage and much more. It is an entry point to the Spark functionality. Thus, it acts a backbone.  
If you have any query about this tutorial, So feel free to Share with us. We will be glad to solve them.