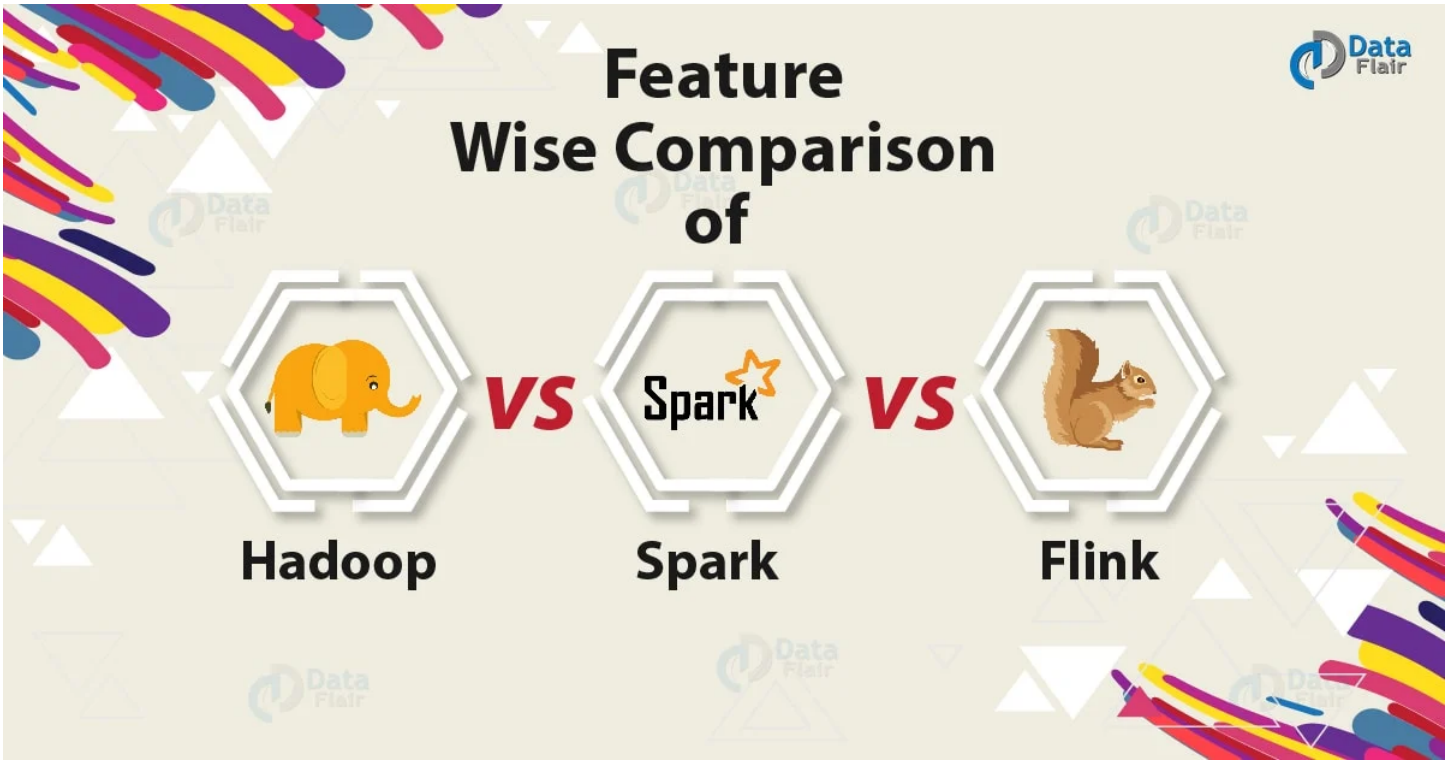
Hadoop vs Spark vs Flink – Big Data Frameworks Comparison

DataFlair Team 6 years ago



Hadoop vs Spark vs Flink

We offer you a brighter future with FREE online courses [Start Now!!]

In this Hadoop vs Spark vs Flink tutorial, we are going to learn feature wise comparison between Apache Hadoop vs Spark vs Flink. These are the top 3 **Big data technologies** that have captured IT market very rapidly with various job roles available for them.

You will understand the limitations of Hadoop for which Spark came into picture and drawbacks of Spark due to which Flink need arose. Here you will learn the difference between Spark and Flink and Hadoop in a detailed manner.

So, let's start Hadoop vs Spark vs Flink.

Comparison between Apache Hadoop vs Spark vs Flink

Before learning the difference between Hadoop vs Spark vs Flink, let us revise the basics of these 3 technologies:
Apache Flink tutorial – 4G of Big Data
Apache Spark tutorial – 3G of Big Data
Big data Hadoop tutorial

So let's start the journey of feature wise comparison between Hadoop vs Spark vs Flink now:

1. Hadoop vs Spark vs Flink – Data Processing

- **Hadoop:** Apache Hadoop built for batch processing. It takes large data set in the input, all at once, processes it and produces the result. Batch processing is very efficient in the processing of high volume data. An output gets delay due to the size of the data and the computational power of the system.
- **Spark**: Apache Spark is also a part of **Hadoop Ecosystem.** It is a batch processing System at heart too but it also supports stream processing.
- **Flink**: Apache Flink provides a single runtime for the streaming and batch processing.

2. Hadoop vs Spark vs Flink – Streaming Engine

- **Hadoop:** Map-reduce is batch-oriented processing tool. It takes large data set in the input, all at once, processes it and produces the result.
- **Spark:** Apache **Spark Streaming** processes data streams in micro-batches. Each batch contains a collection of events that arrived over the batch period. But it is not enough for use cases where we need to process large streams of live data and provide results in real time.
- **Flink:** Apache Flink is the true streaming engine. It uses streams for workloads: streaming, SQL, micro-batch, and batch. Batch is a finite set of streamed data.

3. Hadoop vs Spark vs Flink – Data Flow

- **Hadoop: MapReduce** computation data flow does not have any loops. It is a chain of stages. At each stage, you progress forward using an output of the previous stage and producing input for the next stage.
- **Spark:** Though Machine Learning algorithm is a cyclic data flow, Spark represents it as **(DAG) direct acyclic graph.**
- **Flink:** Flink takes a different approach than others. It supports controlled cyclic dependency graph in run time. This helps to represent the Machine Learning algorithms in a very efficient way.

4. Hadoop vs Spark vs Flink – Computation Model

- **Hadoop:** MapReduce adopted the batch-oriented model. Batch is processing data at rest. It takes a large amount of data at once, processing it and then writing out the output.
- **Spark:** Spark has adopted micro-batching. Micro-batches are an essentially "collect and then process" kind of computational model.
- **Flink:** Flink has adopted a continuous flow, operator-based streaming model. A continuous flow operator processes data when it arrives, without any delay in collecting the data or processing the data.

5. Hadoop vs Spark vs Flink – Performance

- **Hadoop:** Apache Hadoop supports batch processing only. It doesn't process streamed data hence performance is slower when compared Hadoop vs Spark vs Flink.
- **Spark:** Though Apache Spark has an excellent community background and now It is considered as most matured community. But Its stream processing is not much efficient than Apache Flink as it uses micro-batch processing.
- **Flink:** Performance of Apache Flink is excellent as compared to any other data processing system. Apache Flink uses native closed loop iteration operators which make machine learning and graph processing more faster when we compare Hadoop vs Spark vs Flink.

6. Hadoop vs Spark vs Flink – Memory management

- **Hadoop:** It provides configurable Memory management. You can do it dynamically or statically.
- **Spark:** It provides configurable memory management. The latest release of Spark 1.6 has moved towards automating memory management.
- **Flink:** It provides automatic memory management. It has its own memory management system, separate from Java's garbage collector.

7. Hadoop vs Spark vs Flink – Fault tolerance

- **Hadoop:** MapReduce is highly fault-tolerant. There is no need to restart the application from scratch in case of any failure in Hadoop.
- **Spark:** Apache Spark Streaming recovers lost work and with no extra code or configuration, it delivers exactly-once semantics out of the box. Read more about Spark Fault Tolerance.
- **Flink:** The fault tolerance mechanism followed by Apache Flink is based on ***Chandy-Lamport distributed snapshots***. The mechanism is lightweight, which results in maintaining high throughput rates and provide strong consistency guarantees at the same time.

8. Hadoop vs Spark vs Flink – Scalability

- **Hadoop:** MapReduce has incredible scalability potential and has been used in production on tens of thousands of Nodes.
- **Spark:** It is highly scalable, we can keep adding n number of nodes in the cluster. A large known sSpark cluster is of 8000 nodes.
- **Flink:** Apache Flink is also highly scalable, we can keep adding n number of nodes in the cluster A large known Flink cluster is of thousands of nodes.

9. Hadoop vs Spark vs Flink – Iterative Processing

- **Hadoop:** It does not support iterative processing.
- **Spark:** It iterates its data in batches. In Spark, each iteration has to be scheduled and executed separately.
- **Flink:** It iterates data by using its streaming architecture. Flink can be instructed to only process the parts of the data that have

actually changed, thus significantly increasing the performance of the job.

10. Hadoop vs Spark vs Flink – Language Support

- **Hadoop:** It Supports Primarily *Java*, other languages supported are *c, c++, ruby, groovy, Perl, Python.*
- **Spark:** It supports *Java, Scala, Python* and *R*. Spark is implemented in Scala. It provides API in other languages like Java, Python, and R.
- **Flink:** It Supports *Java, Scala, Python and R*. Flink is implemented in Java. It does provide Scala API too.

11. Hadoop vs Spark vs Flink – Optimization

- **Hadoop:** In MapReduce, jobs have to be manually optimized. There are several ways to optimize the MapReduce Jobs: Configure your cluster correctly, use a combiner, use LZO compression, tune the number of MapReduce Task appropriately and use the most appropriate and compact writable type for your data.
- **Spark:** In Apache Spark, jobs have to be manually optimized. There is a new extensible optimizer, **Catalyst**, based on functional programming construct in Scala. Catalyst's extensible design had two purposes: First, easy to add new optimization techniques. Second, enable external developers to extend the optimizer catalyst.
- **Flink:** Apache Flink comes with an optimizer that is independent with the actual programming interface. The Flink optimizer works similarly to a relational Database Optimizer but applies these optimizations to the Flink programs, rather than SQL queries.

12. Hadoop vs Spark vs Flink – Latency

- **Hadoop:** The MapReduce framework of Hadoop is relatively slower since it is designed to support the different format, structure and the huge volume of data. That's why Hadoop has higher latency than both Spark and Flink.
- **Spark:** Apache Spark is yet another batch processing system but it is relatively faster than Hadoop MapReduce since it caches much of the input data on memory by **RDD** and keeps intermediate data

in memory itself, eventually writes the data to disk upon completion or whenever required.
- **Flink:** With small efforts in configuration, Apache Flink's data streaming runtime achieves low latency and high throughput.

13. Hadoop vs Spark vs Flink – Processing Speed

- **Hadoop:** MapReduce processes slower than Spark and Flink. The slowness occurs only because of the nature of the MapReduce-based execution, where it produces lots of intermediate data, much data exchanged between nodes, thus causes huge disk IO latency. Furthermore, it has to persist much data in disk for synchronization between phases so that it can support Job recovery from failures. Also, there are no ways in MapReduce to cache all subset of the data in memory.
- **Spark:** Apache Spark processes faster than MapReduce because it caches much of the input data on memory by RDD and keeps intermediate data **in memory** itself, eventually writes the data to disk upon completion or whenever required. Spark is 100 times faster than MapReduce and this shows how Spark is better than Hadoop MapReduce.
- **Flink:** It processes faster than Spark because of its streaming architecture. Flink increases the performance of the job by instructing to only process part of data that have actually changed.

14. Hadoop vs Spark vs Flink – Visualization

- **Hadoop:** In Hadoop, data visualization tool is **zoomdata** that can connect directly to **HDFS** as well on SQL-on-Hadoop technologies such as Impala, **Hive**, **Spark SQL,** Presto and more.
- **Spark:** It offers a web interface for submitting and executing jobs on which the resulting execution plan can be visualized. Flink and Spark both are integrated to Apache **zeppelin** It provides data analytics, ingestion, as well as discovery, visualization, and collaboration.
- **Flink:** It also offers a web interface for submitting and executing jobs. The resulting execution plan can be visualized on this interface.

15. Hadoop vs Spark vs Flink – Recovery

- **Hadoop:** MapReduce is naturally resilient to system faults or failures. It is the highly fault-tolerant system.
- **Spark:** Apache Spark RDDs allow recovery of partitions on failed nodes by re-computation of the **DAG** while also supporting a more similar recovery style to Hadoop by way of checkpointing, to reduce the dependencies of RDDs.
- **Flink:** It supports checkpointing mechanism that stores the program in the data sources and data sink, the state of the window, as well as user-defined state that recovers streaming job after failure.

16. Hadoop vs Spark vs Flink – Security

- **Hadoop:** It supports **Kerberos authentication**, which is somewhat painful to manage. But, third party vendors have enabled organizations to leverage Active Directory Kerberos and LDAP for authentication.
- **Spark:** Apache Spark's security is a bit sparse by currently only supporting authentication via shared secret (password authentication). The security bonus that Spark can enjoy is that if you run Spark on HDFS, it can use HDFS ACLs and file-level permissions. Additionally, Spark can run on **YARN** to use Kerberos authentication.
- **Flink:** There is user-authentication support in Flink via the Hadoop / Kerberos infrastructure. If you run Flink on YARN, Flink acquires the Kerberos tokens of the user that submits programs, and authenticate itself at YARN, HDFS, and **HBase** with that.Flink's upcoming connector, streaming programs can authenticate themselves as stream brokers via SSL.

17. Hadoop vs Spark vs Flink – Cost

- **Hadoop:** MapReduce can typically run on less expensive hardware than some alternatives since it does not attempt to store everything in memory.
- **Spark:** As spark requires a lot of RAM to run in-memory, increasing it in the cluster, gradually increases its cost.

- **Flink:** Apache Flink also requires a lot of RAM to run in-memory, so it will increase its cost gradually.

18. Hadoop vs Spark vs Flink – Compatibility

- **Hadoop:** Apache Hadoop MapReduce and Apache Spark are compatible with each other and Spark shares all MapReduce's compatibilities for data sources, file formats, and business intelligence tools via JDBC and ODBC.
- **Spark:** Apache Spark and Hadoop are compatible to each other. Spark is compatible with Hadoop data. It can run in Hadoop clusters through YARN or Spark's standalone mode, and it can process data in HDFS, HBase, Cassandra, Hive, and any Hadoop InputFormat.
- **Flink:** Apache Flink is a scalable data analytics framework that is fully compatible to Hadoop. It provides a Hadoop Compatibility package to wrap functions implemented against Hadoop's MapReduce interfaces and embed them in Flink programs.

19. Hadoop vs Spark vs Flink – Abstraction

- **Hadoop:** In MapReduce, we don't have any type of abstraction.
- **Spark:** In Spark, for batch, we have *Spark RDD* abstraction and *DStream* for streaming which is internally RDD itself.
- **Flink:** In Flink, we have Dataset abstraction for batch and DataStreams for the streaming application.

20. Hadoop vs Spark vs Flink – Easy to use

- **Hadoop:** MapReduce developers need to hand code each operation which makes it very difficult to work.
- **Spark:** It is easy to program as it has tons of high-level operators.
- **Flink:** It also has high-level operators.

21. Hadoop vs Spark vs Flink – Interactive Mode

- **Hadoop:** MapReduce does not have interactive Mode.
- **Spark:** Apache Spark has an interactive shell to learn how to make the most out of Apache Spark. This is a Spark application written in Scala to offer a command-line environment with auto-

completion where you can run ad-hoc queries and get familiar with the features of Spark.
- **Flink:** It comes with an integrated interactive Scala Shell. It can be used in a local setup as well as in a cluster setup.

22. Hadoop vs Spark vs Flink – Real-time Analysis

- **Hadoop:** MapReduce fails when it comes to real-time data processing as it was designed to perform batch processing on voluminous amounts of data.
- **Spark:** It can process real time data ie data coming from the real-time event streams at the rate of millions of events per second.
- **Flink:** It is mainly used for real-time data Analysis Although it also provides fast batch data Processing.

23. Hadoop vs Spark vs Flink – Scheduler

- **Hadoop:** Scheduler in Hadoop becomes the pluggable component. There are two schedulers for multi-user workload: *Fair Scheduler a*nd *Capacity Scheduler*. To schedule complex flows, MapReduce needs an external job scheduler like **Oozie**.
- **Spark:** Due to in-memory computation, spark acts its own flow scheduler.
- **Flink:** Flink can use YARN Scheduler but Flink also has its own Scheduler.

24. Hadoop vs Spark vs Flink – SQL support

- **Hadoop:** It enables users to run SQL queries using Apache Hive.
- **Spark:** It enables users to run SQL queries using Spark-SQL. Spark provides both Hives like query language and **Dataframe** like DSL for querying structured data.
- **Flink:** In Flink, Table API is an SQL-like expression language that supports data frame like DSL and it's still in beta. There are plans to add the SQL interface but not sure when it will land in the framework.

25. Hadoop vs Spark vs Flink – Caching

- **Hadoop:** MapReduce cannot cache the data in memory for future requirements
- **Spark:** It can cache data in memory for further iterations which enhance its performance.
- **Flink:** It can cache data in memory for further iterations to enhance its performance.

26. Hadoop vs Spark vs Flink – Hardware Requirements

- **Hadoop:** MapReduce runs very well on Commodity Hardware.
- **Spark:** Apache Spark needs mid to high-level hardware. Since Spark cache data in-memory for further iterations which enhance its performance.
- **Flink:** Apache Flink also needs mid to High-level Hardware. Flink can also cache data in memory for further iterations which enhance its performance.

27. Hadoop vs Spark vs Flink – Machine Learning

- **Hadoop:** It requires machine learning tool like **Apache Mahout**.
- **Spark:** It has its own set of machine learning MLlib. Within memory caching and other implementation details, it's really powerful platform to implement ML algorithms.
- **Flink:** It has FlinkML which is Machine Learning library for Flink. It supports controlled *cyclic dependency graph* in runtime. This makes them represent the ML algorithms in a very efficient way compared to *DAG* representation.

28. Hadoop vs Spark vs Flink – Line of code

- **Hadoop:** Hadoop 2.0 has 1,20,000 line of codes. More no of lines produce more no of bugs and it will take much time to execute the program.
- **Spark:** Apache Spark is developed in merely 20000 line of codes. No. of the line of code is lesser than Hadoop. So it will take less time to execute the program.
- **Flink:** Flink is developed in scala and java, so no. of the line of code is lesser than Hadoop. So it will also take the less time to execute the program.

29. Hadoop vs Spark vs Flink – High Availability

The High availability refers to a system or component that is operational for long length of time.

- **Hadoop:** Configurable in High Availability Mode.
- **Spark:** Configurable in High Availability Mode.
- **Flink:** Configurable in High Availability Mode.

30. Hadoop vs Spark vs Flink – Amazon S3 connector

Amazon Simple Storage Service (Amazon S3) is object storage with a simple web service interface to store and retrieve any amount of data from anywhere on the web.

- **Hadoop:** Provides Supports for Amazon S3 Connector.
- **Spark:** Provides Supports for Amazon S3 Connector.
- **Flink:** Provides Supports for Amazon S3 connector.

31. Hadoop vs Spark vs Flink – Deployment

- **Hadoop:** In Standalone mode, Hadoop is configured to run in a single-node, non-distributed mode. In Pseudo-Distributed mode, Hadoop runs in a pseudo distributed mode. Thus, the difference is that each Hadoop daemon runs in a separate Java process in pseudo-distributed mode. Whereas in local mode each Hadoop daemon runs as a single Java process. In a fully-distributed mode, all daemons execute in separate nodes forming a multi-node cluster.
- **Spark:** It also provides a simple standalone deploy mode to running on the **Mesos** or YARN cluster managers. It can be launched either manually, by starting a master and workers by hand or use our provided launch scripts. It is also possible to run these daemons on a single machine for testing.
- **Flink:** It also provides standalone deploy mode to running on YARN cluster Managers.

32. Hadoop vs Spark vs Flink – Back pressure Handing

BackPressure refers to the buildup of data at an I/O switch when buffers are full and not able to receive more data. No more data packets transfer until the bottleneck of data eliminates or the buffer is empty.

- **Hadoop:** It handles back pressure through Manual Configuration.
- **Spark:** It also handles back pressure through Manual Configuration.
- **Flink:** It handles back pressure Implicitly through System Architecture.

33. Hadoop vs Spark vs Flink – Duplication Elimination

- **Hadoop:** There is no duplication elimination in Hadoop.
- **Spark:** Spark also processes every record exactly one time hence eliminates duplication.
- **Flink:** Apache Flink processes every record exactly one time hence eliminates duplication. Streaming applications can maintain custom state during their computation. Flink's checkpointing mechanism ensures exactly once semantics for the state in the presence of failures.

34. Hadoop vs Spark vs Flink – Windows criteria

A data stream needs to be grouped into many logical streams on each of which a window operator can be applied.

- **Hadoop:** It doesn't support streaming so there is no need of window criteria.
- **Spark:** It has time-based window criteria.
- **Flink:** It has record-based or any custom user-defined Flink Window criteria.

35. Hadoop vs Spark vs Flink – Apache License

The Apache License, Version 2.0 (ALv2) is a permissive free software license written by the Apache Software Foundation (ASF). The Apache License requires preservation of the copyright notice and disclaimer.

- **Hadoop:** Apache License 2.
- **Spark:** Apache License 2.
- **Flink:** Apache License 2.

So, this is how the comparison is done between the top 3 Big data technologies Hadoop vs Spark vs Flink.