Q/A

1. How does Apache flink different from the Apache Hadoop and Spark?

Ans: Hadoop is mainly designed for batch processing which is very efficient in processing large datasets. Similarly Apache Spark can process the streaming data but its streaming is micro-batch streaming i.e it provides only the near time processing but Apache flink supports both batch processing as well as stream processing in single engine.
Some other difference are tabulated below:

Based On	Hadoop	Spark	Flink
Data Processing	Hadoop is mainly designed for batch processing which is very efficient in processing large datasets.	It supports batch processing as well as stream processing.	It supports both batch and stream processing. Flink also provides the single run-time for batch and stream processing.
Stream Engine	It takes the complete data-set as input at once and produces the output.	Process data streams in micro-batches.	The true streaming engine uses streams for workload: streaming, micro-batch, SQL, batch.
Data Flow	Data Flow does not contain any loops. supports linear data flow.	Spark supports cyclic data flow and represents it as (DAG) direct acyclic graph.	Flink uses a controlled cyclic dependency graph in run time. which efficiently manifest ML algorithms.
Computation Model	Hadoop Map-Reduce supports the batch-oriented model.	It supports the micro-batching computational model.	Flink supports a continuous operator-based streaming model.
Fault tolerance	Highly fault-tolerant using a replication mechanism.	Spark RDD provides fault tolerance through lineage.	Fault tolerance is based on Chandy-Lamport distributed snapshots which results in high throughput.

Apache Flink Summary:

Flink is a distributed processing engine and a scalable data analytics framework. You can use Flink to process data streams at a large scale and to deliver real-time analytical insights about your processed data with your streaming application.

Flink is designed to run in all common cluster environments, perform computations at in-memory speed and at any scale. Furthermore, Flink provides communication, fault tolerance, and data distribution for distributed computations over data streams.

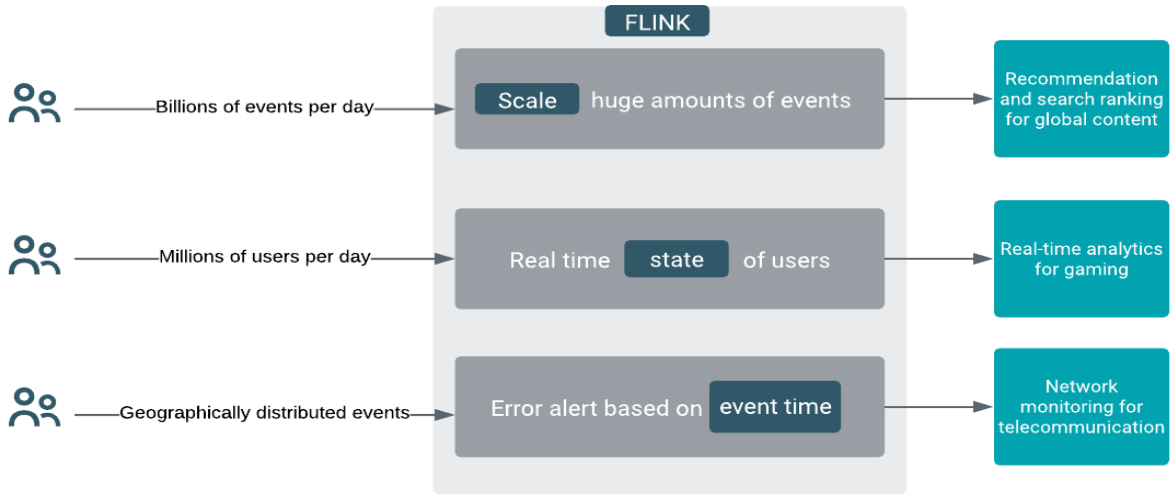
Flink applications process stream of events as unbounded or bounded data sets. Unbounded streams have no defined end and are continuously processed. Bounded streams have an exact start and end, and can be processed as a batch. In terms of time, Flink can process real-time data as it is generated and stored data in storage filesystems. In CSA, Flink is used for unbounded, real-time stream processing.

Streaming use cases with Flink

You can create your Flink streaming applications based on the role of your processed data. Flink use cases include fraud detection, network monitoring, alert triggering, and other solutions to enhance user experience.

A large variety of enterprises choose Flink as a stream processing platform due to its ability to handle scale, stateful stream processing, and event time. See the following

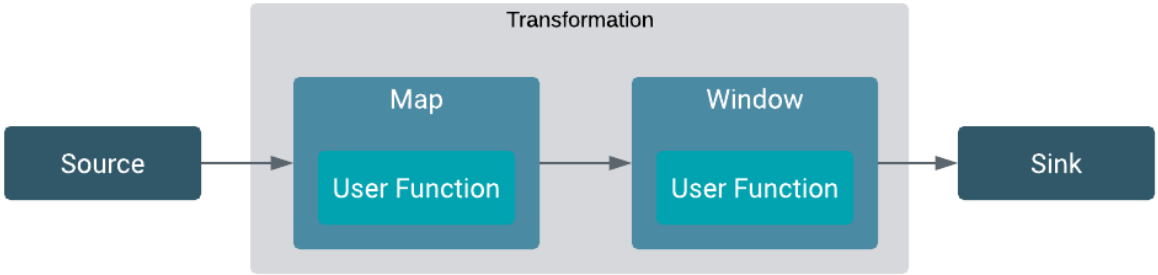
illustration for example use cases.



Flink Streaming Applications

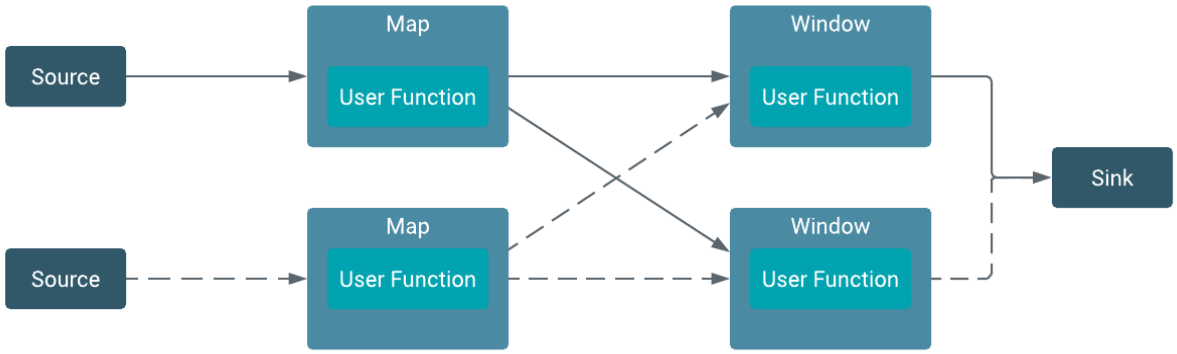
You can use APIs to develop Flink streaming applications where the data pipeline consists of one or more data source, data transformation, and data sink. You can build the architecture of your application with parallelism and windowing functions to benefit from the scalability and state handling features of Flink.

The DataStream API is used as the core API to develop Flink streaming applications using Java or Scala programming languages. The core building blocks of a streaming application are datastream and transformation. In a Flink program, the incoming data streams from a source are transformed by a defined operation which results in one or more output streams to the sink as shown in the following illustration.



The structure of this dataflow is implemented in a pipeline that gives a Flink application its core logic. On a dataflow one or more operations can be defined which can be

processed in parallel and independently to each other. With windowing functions, different computations can be applied to different streams in the defined time window to further maintain the processing of events. The following image illustrates the parallel structure of dataflows.



Handling state in Flink

You can use Flink to store the state of your application locally in state backends that guarante lower latency when accessing your processed data. You can also create checkpoints and savepoints to have a fault-tolerant backup of your streaming application on a durable storage.

Event-driven applications with Flink

In time-sensitive cases where the application uses alerting or triggering functions, it is important to distinguish between event time and processing time. To make the designing of applications easier, you can create your Flink application either based on the time when the event is created or when it is processed by the operator.

Sophisticated windowing in Flink

The windowing feature of Flink helps you to determine different time sections of your unbounded data streams. This way you can avoid missing events that arrive late and you can easily apply different transformations on your streaming data.

Using watermark in Flink

For a streaming application of unbounded data sets, the completeness of all incoming data is crucial. To guarantee that every data is processed, you can use watermarks in Flink applications to track the progress of time for events.

Creating checkpoints and savepoints in Flink

You can use checkpoints and savepoints to make Flink applications fault tolerant throughout the whole pipeline. With checkpoints and savepoints, you can create a backup mechanism from which you can restore your whole application, with or without state, in case of failure or upgrade.