Apache Flink®

Stateful Computations over Data Streams

Apache Flink is a framework and distributed processing engine for stateful computations over unbounded and bounded data streams. Flink has been designed to run in all common cluster environments, perform computations at in-memory speed and at any scale.

What is Apache Flink? — Architecture

Apache Flink is a framework and distributed processing engine for stateful computations over *unbounded and bounded* data streams. Flink has been designed to run in *all common cluster environments*, perform computations at *in-memory speed* and at *any scale*.

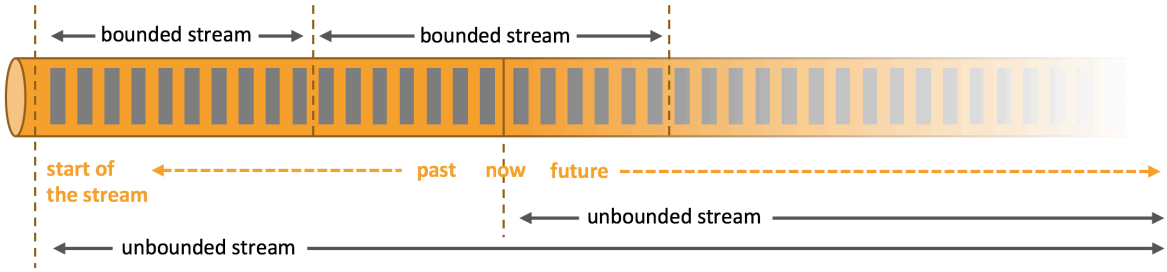
Here, we explain important aspects of Flink's architecture.

Process Unbounded and Bounded Data

Any kind of data is produced as a stream of events. Credit card transactions, sensor measurements, machine logs, or user interactions on a website or mobile application, all of these data are generated as a stream.

Data can be processed as *unbounded* or *bounded* streams.

1. **Unbounded streams** have a start but no defined end. They do not terminate and provide data as it is generated. Unbounded streams must be continuously processed, i.e., events must be promptly handled after they have been ingested. It is not possible to wait for all input data to arrive because the input is unbounded and will not be complete at any point in time. Processing unbounded data often requires that events are ingested in a specific order, such as the order in which events occurred, to be able to reason about result completeness.
2. **Bounded streams** have a defined start and end. Bounded streams can be processed by ingesting all data before performing any computations. Ordered ingestion is not required to process bounded streams because a bounded data set can always be sorted. Processing of bounded streams is also known as batch processing.



Apache Flink excels at processing unbounded and bounded data sets. Precise control of time and state enable Flink's runtime to run any kind of application on unbounded streams. Bounded streams are internally processed by algorithms and data structures that are specifically designed for fixed sized data sets, yielding excellent performance.

Convince yourself by exploring the use cases that have been built on top of Flink.

Deploy Applications Anywhere

Apache Flink is a distributed system and requires compute resources in order to execute applications. Flink integrates with all common cluster resource managers such as Hadoop YARN and Kubernetes but

can also be setup to run as a stand-alone cluster.

Flink is designed to work well each of the previously listed resource managers. This is achieved by resource-manager-specific deployment modes that allow Flink to interact with each resource manager in its idiomatic way.

When deploying a Flink application, Flink automatically identifies the required resources based on the application's configured parallelism and requests them from the resource manager. In case of a failure, Flink replaces the failed container by requesting new resources. All communication to submit or control an application happens via REST calls. This eases the integration of Flink in many environments.

Run Applications at any Scale

Flink is designed to run stateful streaming applications at any scale. Applications are parallelized into possibly thousands of tasks that are distributed and concurrently executed in a cluster. Therefore, an application can leverage virtually unlimited amounts of CPUs, main memory, disk and network IO. Moreover, Flink easily maintains very large application state. Its asynchronous and incremental checkpointing algorithm ensures minimal impact on processing latencies while guaranteeing exactly-once state consistency.

Users reported impressive scalability numbers for Flink applications running in their production environments, such as

- applications processing **multiple trillions of events per day**,
- applications maintaining **multiple terabytes of state**, and
- applications **running on thousands of cores**.

Leverage In-Memory Performance

Stateful Flink applications are optimized for local state access. Task state is always maintained in memory or, if the state size exceeds the available memory, in access-efficient on-disk data structures. Hence, tasks perform all computations by accessing local, often in-memory, state yielding very low processing latencies. Flink guarantees exactly-once state consistency in case of failures by periodically and asynchronously checkpointing the local state to durable storage.