

Flink — Stream and Batch Processing in a Single Engine

Prepared by : Hagar Muhamed

Lecture 8





مجموعة جامعة
الجميع الذكية
للتعليم القابضة



Flink — Stream and Batch Processing in a Single Engine

Lecture 8

Prepared by
Dr. Hagar Muhamed

www.midocean.ae

Flink — Stream and Batch Processing in a Single Engine

- Flink Principles and Architecture
- ▣ Flink Time and Window
- ▣ Flink Watermark
- ▣ Flink Fault Tolerance Mechanism

Chapter **08**

Content



What Is Flink?

In the current generation, Apache Flink is the big giant tool that is a 4G of Big Data. It's the true stream processing framework. Flink's kernel (core) is a streaming runtime that provides distributed processing, fault tolerance. Flink processes events at a constantly high speed with low latency. It schemes the data at lightning-fast speed. Apache Flink is the large-scale data processing framework that we can reuse when data is generated at high velocity.

This is an important open-source platform that can address numerous types of conditions efficiently:

1. Batch Processing
2. Iterative Processing
3. Real-time stream processing
4. Interactive processing
5. In-memory processing
6. Graph Processing

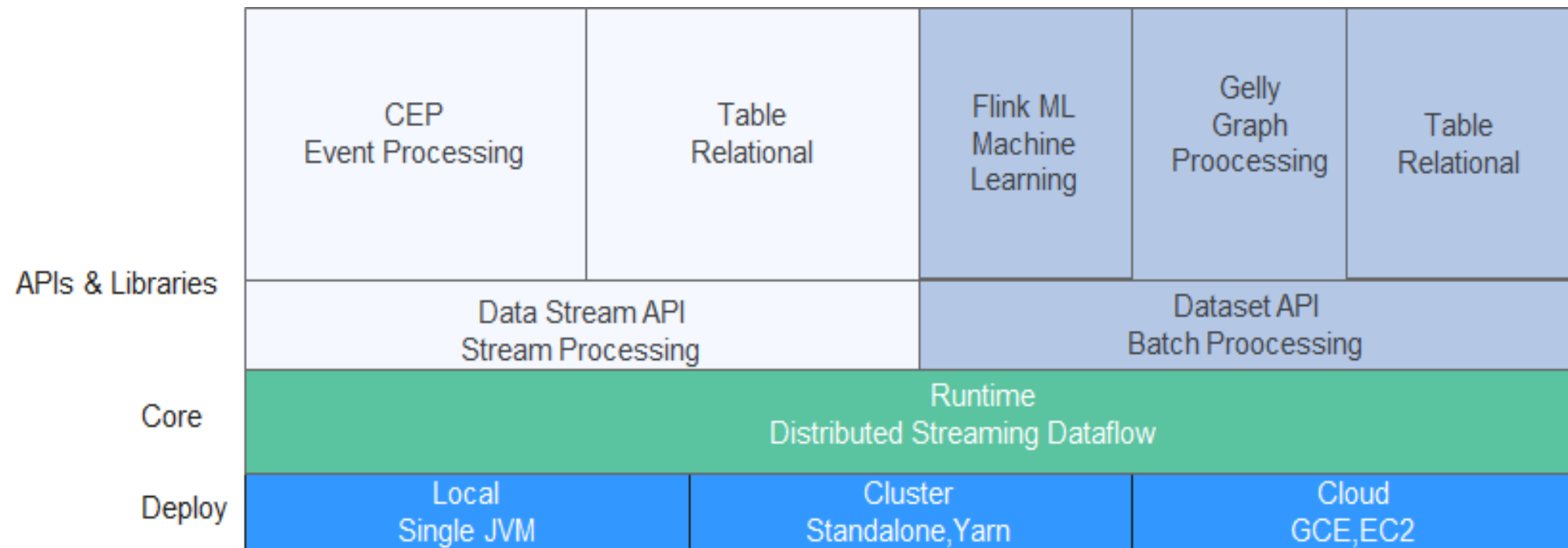
Why Apache Flink?

Flink is volition to MapReduce, it processes data further than 100 times faster than MapReduce. It's independent of Hadoop but it can use HDFS to read, write, store, process the data. Flink doesn't give its own data storehouse system. It takes data from the distributed storage system. The crucial vision for Apache Flink is to overcome and reduce the complexity that has been faced by other distributed data-driven machines.

Chapter

08

Flink Runtime Architecture



Core Concept — DataStream

Flink uses class `DataStream` to represent the stream data in Flink programs. You can think of `DataStream` as immutable collections of data that can contain duplicates. The number of elements in `DataStream` is unlimited.

Core Concept — DataStream

Operator operations between DataStreams:

Window operations are associated using the **reduce**, **sum** and **max** functions.

Connect: Connects streams using the **flat map** and **map** functions.

Joined Stream: Performs the join operation between streams using the **join** function, which is similar to the join operation in a database.

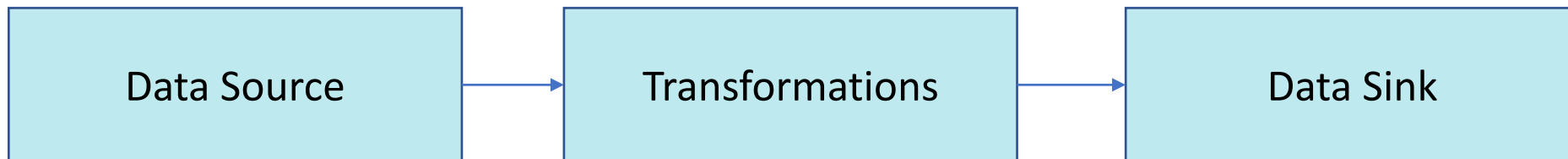
Keyed Stream: Processes data flows based on keys using the **keyby** function.

Core Concept — DataSet

DataSet programs in Flink are regular programs that transform data sets (for example, filtering, mapping, joining, and grouping). The datasets are initially created by reading files or from local collections. Results are returned via sinks, which can write the data to (distributed) files or to standard output (for example, the command line terminal).

Flink Program

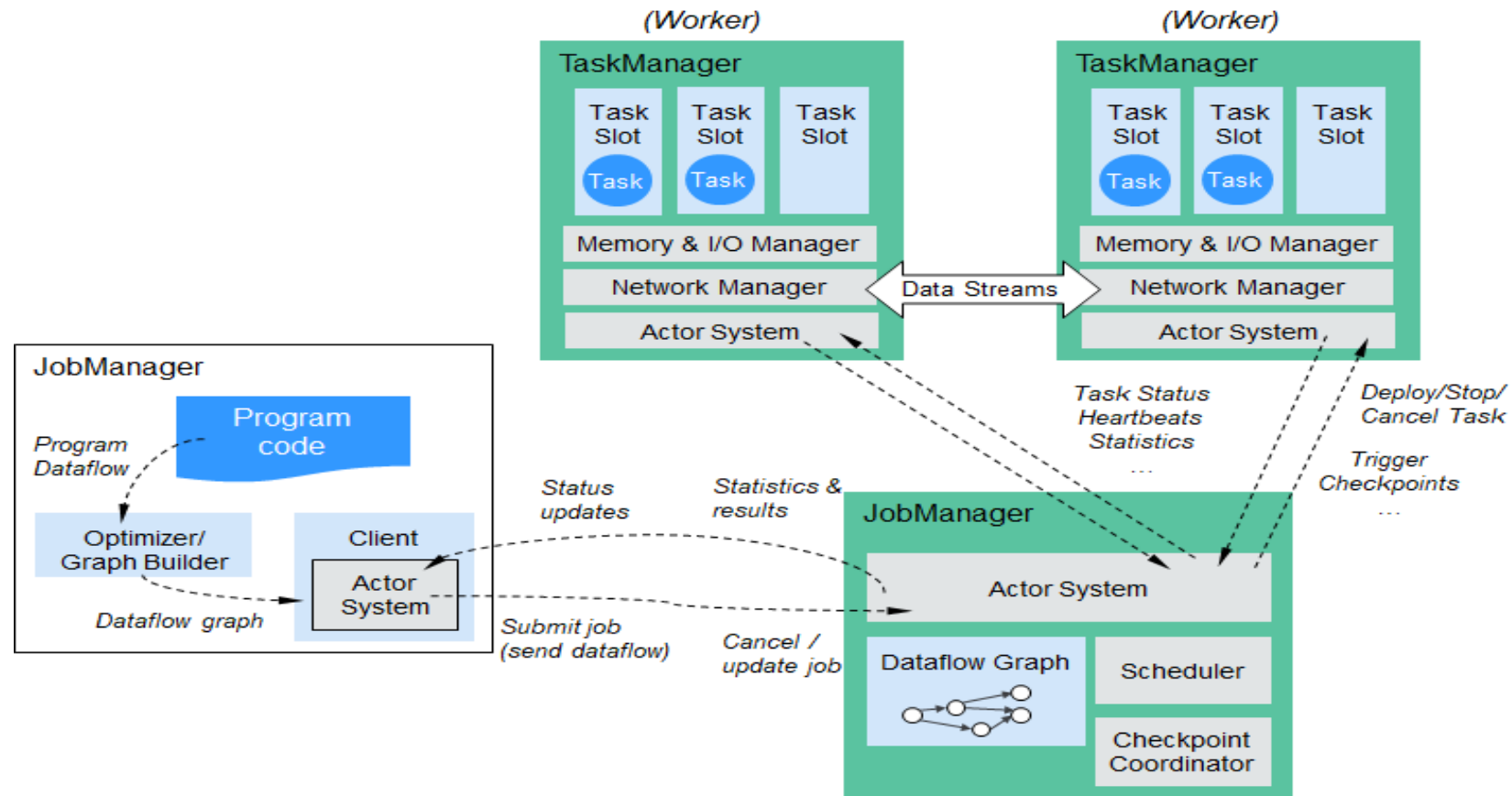
- Flink programs consist of three parts: source, transformation, and sink. The source reads data from data sources such as HDFS, Kafka, and text. The transformation is responsible for data transformation operations. The sink outputs final data (such as HDFS, Kafka, and text). Data that flows between these parts is called a stream.



Chapter

08

Flink Program Running



Flink Program Running

Flink adopts a master-slave architecture. If a Flink cluster is started, a Job Manager process and at least one Task Manager process are also started.

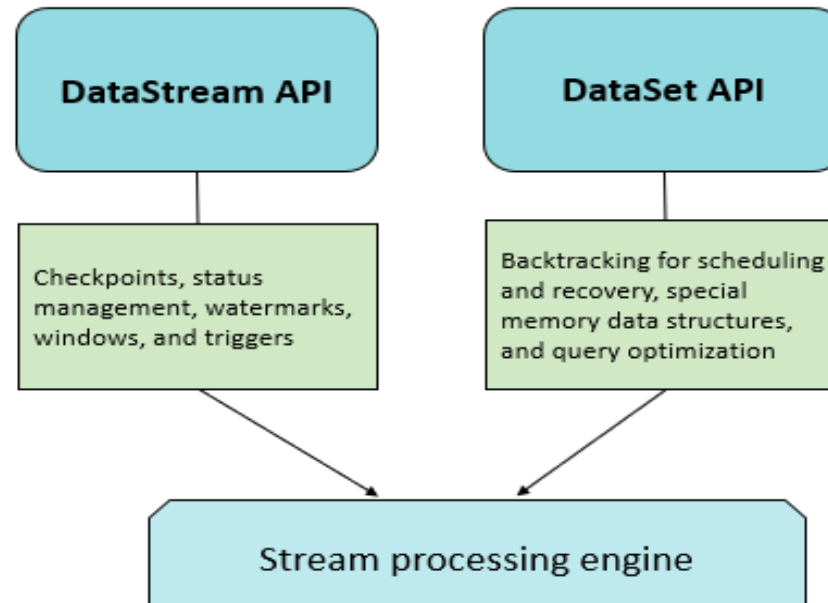
After a Flink program is submitted, a client is created for preprocessing. The program is converted into a DAG that indicates a complete job and is submitted to Job Manager.

Flink Program Running

Job Manager assigns each task in the job to Task Manager. The computing resources in Flink are defined by task slots. Each task slot represents a fixed-size resource pool of Task Manager. For example, a Task Manager with three slots evenly divides the memory managed by the Task Manager into three parts and allocates them to each slot. Slotting resources means that tasks from different jobs do not compete for memory. Currently, slots support only memory isolation but not CPU isolation.

Flink Batch Processing Model

- Flink uses an underlying engine to support both stream processing and batch processing.



Chapter

08

- Based on the stream processing engine, Flink has the following mechanisms:
 - Checkpoint and state mechanisms: used to implement fault tolerance and stateful processing.
 - Watermark mechanism: used to implement the event clock.
 - Window and trigger: used to limit the calculation scope and define the time when the results are displayed.

Chapter

08

- On the same stream processing engine, Flink has another mechanism to implement efficient batch processing.
 - Backtracking for scheduling and recovery: introduced by Microsoft Dryad and now used in almost all batch processors.
 - Special memory data structure used for hashing and sorting: Part of the data can be allowed to flow from the memory to the hard disk when necessary.
 - Optimizer: shortens the time for generating results as much as possible.

Flink — Stream and Batch Processing in a Single Engine

- Flink Principles and Architecture
- ▣ Flink Time and Window
- ▣ Flink Watermark
- ▣ Flink Fault Tolerance Mechanism

Chapter **08**

Content



Time Background

- In stream processor programming, time processing is very critical. For example, event stream data (such as server log data, web page click data, and transaction data) is continuously generated. In this case, you need to use keys to group events and count the events corresponding to each key at a specified interval. This is our known "big data" application.

Time Classification in Stream Processing

- During data stream processing, the system time, that is, processing time, is often used as the time of an event. In fact, the processing time is the time imposed on the event. Due to reasons such as network latency, the processing time cannot reflect the sequence of events.
- In actual scenarios, the time of each event can be classified into three types:
 - Event time: the time when an event occurs.
 - Ingestion time: the time when an event arrives at the stream processing system.
 - Processing time: the time when an event is processed by the system.

Example Time Classifications

For example, if a log enters Flink at 10:00:00.123 on November 12, 2019 and reaches Windows at 10:00:01.234 on November 12, 2019, the log content is as follows:

2019-11-02 18:37:15.624 INFO Fail over to rm2

2019-11-02 18:37:15.624 is the event time.

2019-11-12 10:00:00.123 is the ingestion time.

2019-11-12 10:00:01.234 is the processing time

Window Overview

Streaming computing system is a data processing engine designed to process infinite data sets. Infinite data sets refer to constantly growing data sets. Window is a method for splitting infinite data sets into finite blocks for processing.

Windows are at the heart of processing infinite streams. Windows split the stream into "buckets" of finite sizes, over which we can apply computations.

Windows Types

Windows can be classified into the following types:

Count window: A data-driven window is generated based on the specified number of data records, which is irrelevant to time.

Time window: A time-driven window is generated based on time.

- Apache Flink is a distributed computing framework that supports infinite stream processing. In Flink, windows can divide infinite streams into finite streams. Flink windows can be either Time windows or Count windows.

Windows Types

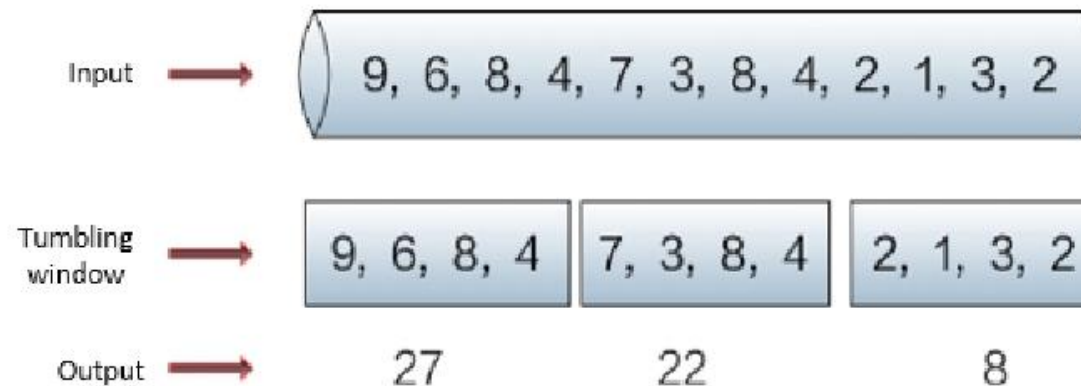
- According to the window implementation principles, time windows can be classified into:
 - Tumbling window
 - Sliding window
 - Session window

Chapter

08

Tumbling window

- The data is sliced according to a fixed window length.
- Characteristics: The time is aligned, the window length is fixed, and the windows do not overlap.
- Use case: BI statistics collection (aggregate calculation in each time period)

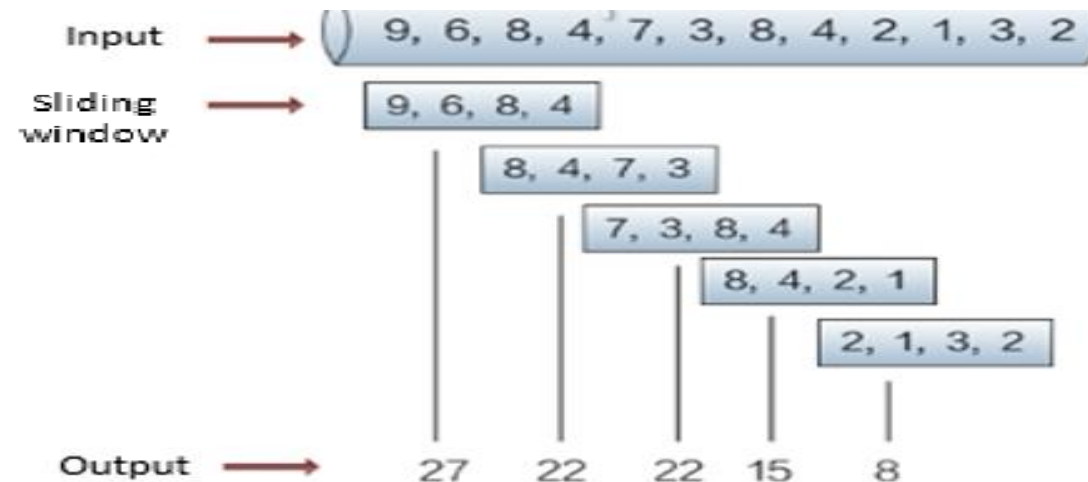


Chapter

08

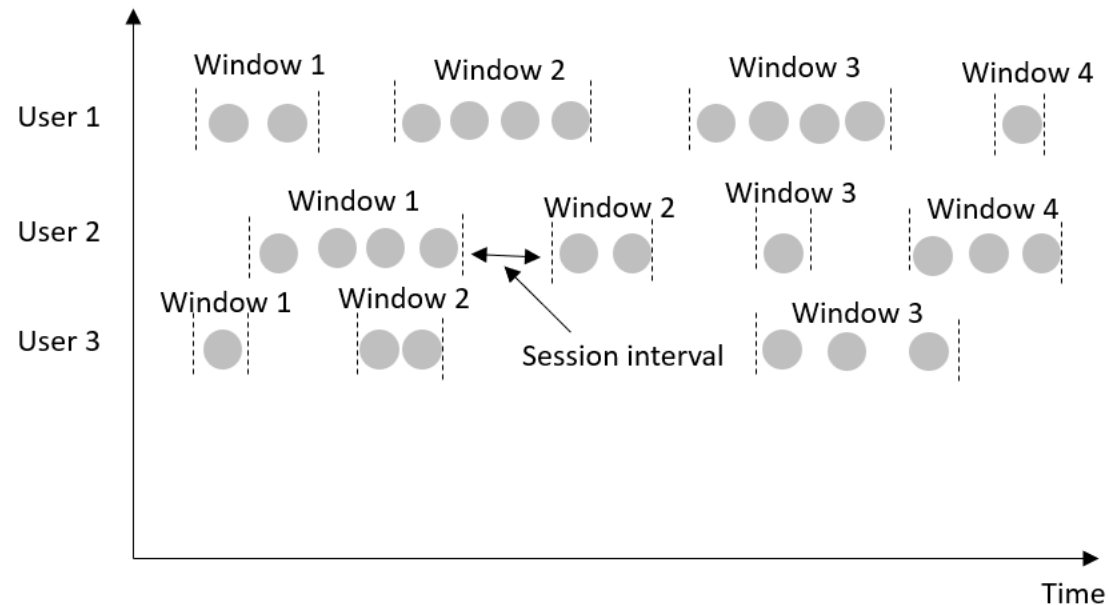
Sliding window

- The sliding window is a more generalized form of a fixed window. It consists of a fixed window length and a sliding interval.
- Characteristics: The time is aligned, the window length is fixed, and the windows can be overlapping.
- Use case: Collect statistics on the failure rate of an API in the last 5 minutes to determine whether to report an alarm.



Session Window

- A session window consists of a series of events and a timeout interval of a specified duration. It is similar to a web application session. That is, a new window is generated if no new data is received within a period of time. Characteristics: The time is not aligned.



Flink — Stream and Batch Processing in a Single Engine

- Flink Principles and Architecture
- ▣ Flink Time and Window
- ▣ Flink Watermark
- ▣ Flink Fault Tolerance Mechanism

Chapter **08**

Content



Out-of-Order Data Problem

- There is a process from event generation to stream processing through the source and then to the operator. In most cases, data is sent to the operator based on the time when the event is generated. However, out-of-order data may be generated, for example, events received by Flink are not sorted strictly based on the event time due to network problems.



Why Are Watermarks Needed?

- For infinite datasets, there is no effective way to determine data integrity. Therefore, watermark is a concept based on event time to describe the integrity of data streams. If events are measured by processing time, everything is ordered and perfect. Therefore, watermarks are not required. In other words, event time causes disorder.

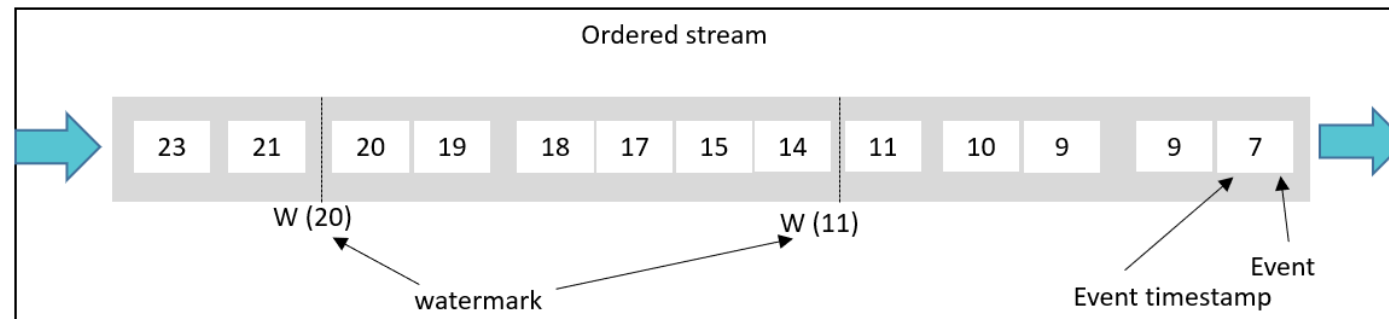
Why Are Watermarks Needed?

- Watermarks are used to solve the disorder problem. The so-called disorder is actually an event delay. For a delayed element, it is impossible to wait indefinitely. A mechanism must be provided to ensure that the window is triggered for calculation after a specific point in time. This special mechanism is watermark, which tells operators that delayed messages should no longer be received.

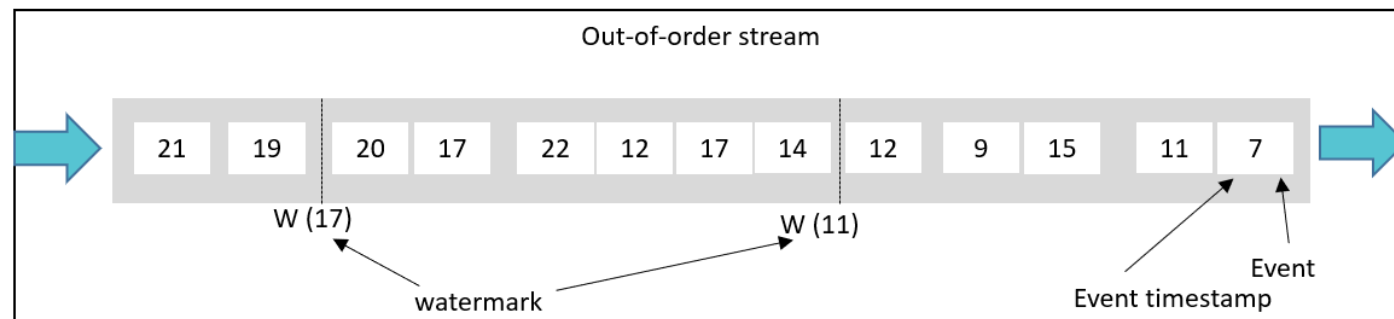
Chapter
08

Watermark Principles

- The following figure shows the watermark of ordered streams (Watermark is set to 0).



- The following figure shows the watermark of out-of-order streams (Watermark is set to 2).



Flink — Stream and Batch Processing in a Single Engine

- Flink Principles and Architecture
- ▣ Flink Time and Window
- ▣ Flink Watermark
- ▣ Flink Fault Tolerance Mechanism

Chapter **08**

Content



Checkpointing

Flink uses a feature called checkpoint to reset the system to the correct state when a fault occurs.

Flink offers the fault tolerance mechanism based on snapshot technology for distributed data streams. It can globally take snapshots of the state data of tasks or operations at the same time point, that is, backs up the states of programs.

Checkpoint Mechanism

- Flink provides the checkpoint fault tolerance mechanism. The distributed snapshot function can take global snapshots of task/operator status data at the same time point in a unified manner. Flink periodically generates checkpoint barriers on the input data set and divides the data within the interval to the corresponding checkpoints through barriers. When an exception occurs in an application, the operator can restore the status of all operators from the previous snapshot to ensure data consistency.

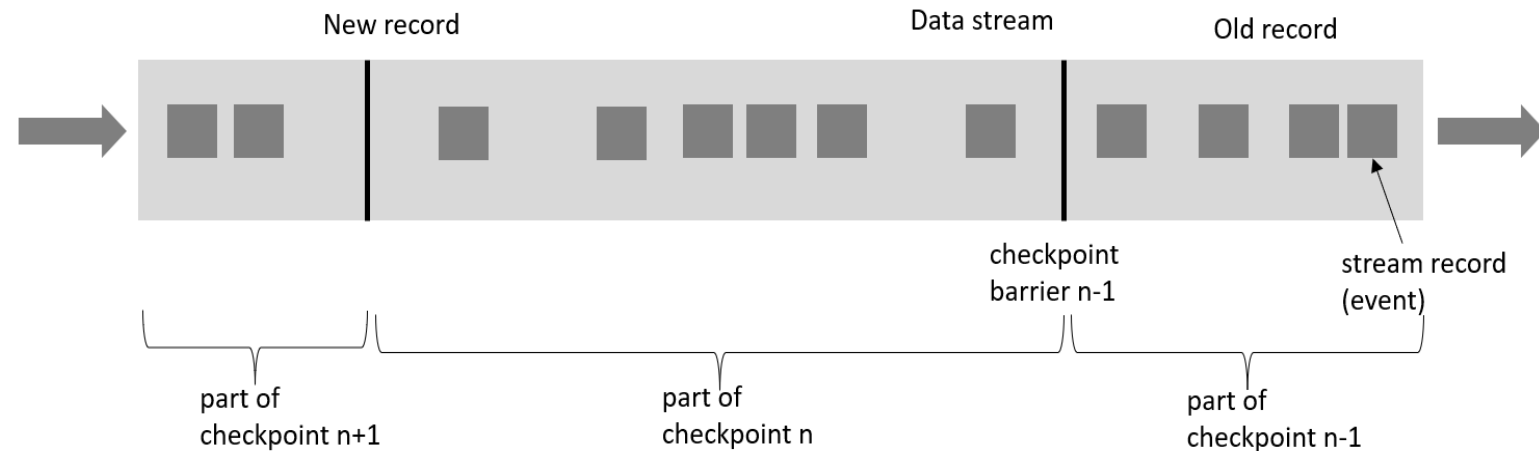
Checkpoint Mechanism

For stream processing applications, these snapshots are very lightweight and can be taken frequently without compromising performance.

For applications with small state, these snapshots are very lightweight and can be taken frequently without impacting the performance much. During checkpointing, the state is stored at a configurable place (such as the JobManager node or HDFS).

Checkpoint Mechanism

- A core element in Flink's distributed snapshotting is stream barriers. These barriers are injected into the data stream and flow with the records as part of the data stream. Barriers flow strictly in line. A barrier separates the records in the data stream into the set of records that goes into the current snapshot, and the records that go into the next snapshot. Barriers do not interrupt the flow of the stream and are hence very lightweight. Multiple barriers from different snapshots can be in the stream at the same time, which means that various snapshots may happen concurrently.



Savepoint vs. Checkpoint

Parameter	Checkpoint	Savepoint
Trigger and Management Method	Automatically triggered and managed by Flink	Manually triggered and managed by users
Usage	Quickly recovers tasks when exceptions occur, for example, timeout exceptions caused by network jitter.	Backs up data as planned, for example, modifying code and adjusting concurrency.
Characteristics	• Lightweight • Automatic restoration from faults • Cleared by default after a job is stopped	• Persistent • Stored in a standard format, allowing code or configuration changes • Manually triggered for restoration from a savepoint

Summary:

Flink is an open-source stream processing and batch processing framework designed for handling large-scale data processing tasks.

It is known for its low-latency, high-throughput stream processing capabilities, making it suitable for real-time data analytics and event-driven applications.

Flink's architecture consists of three layers: core, API, and libraries, allowing developers to create complex data processing pipelines with ease.

It supports various data sources like HDFS, Kafka, and text, and offers a wide range of transformation and aggregation functions for data manipulation.

Flink is a powerful tool for businesses and organizations seeking to harness the potential of big data and make informed decisions in real-time.



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY



Thank You

