Flink Training for Real-Time Data Engineering

Working with Windows in Apache Flink

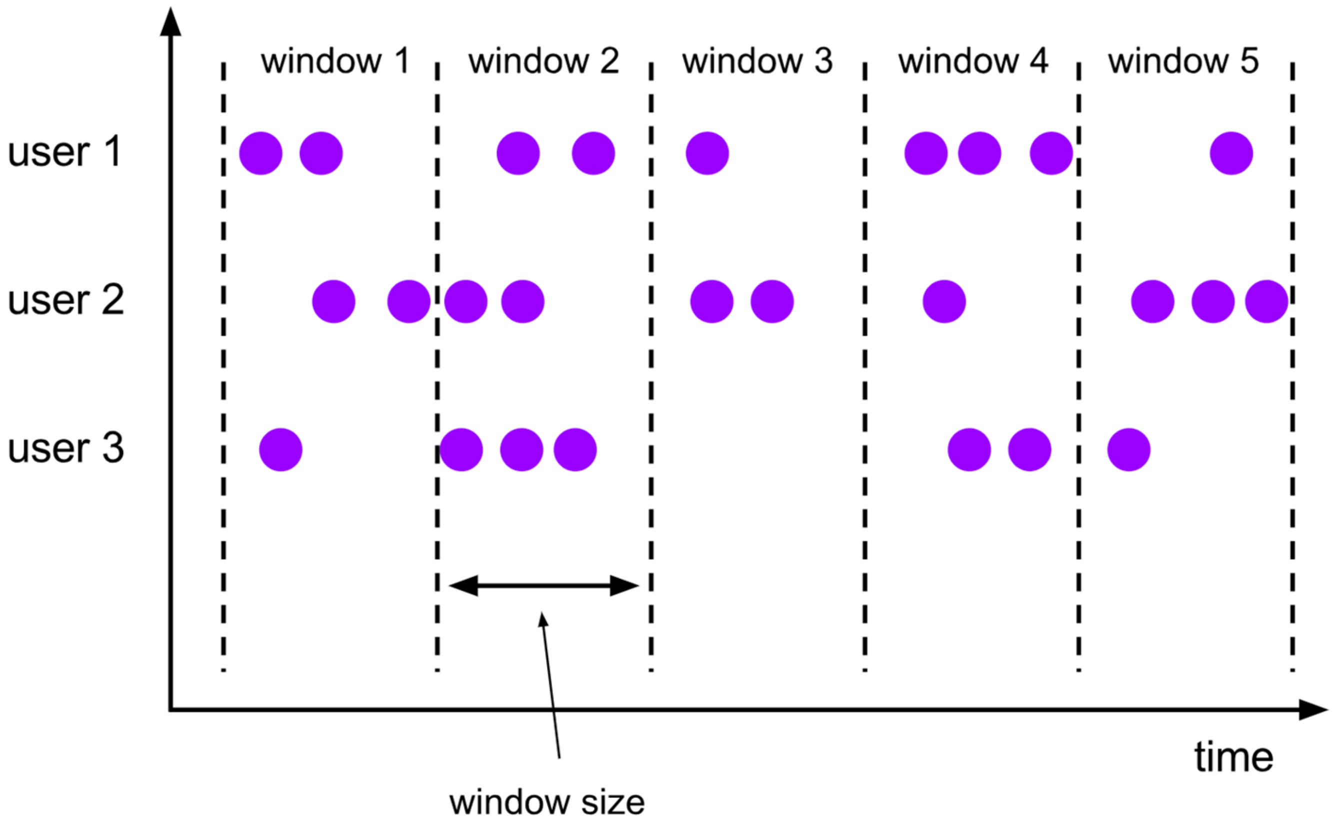
AGENDA

- Time-based windows: Tumbling, Sliding, Session
- Count windows and global windows
- Triggers, evictors, and late data handling
- Real-world use cases: User activity aggregation, fraud detection

Time-based windows: Tumbling, Sliding, Session

Tumbling Windows:

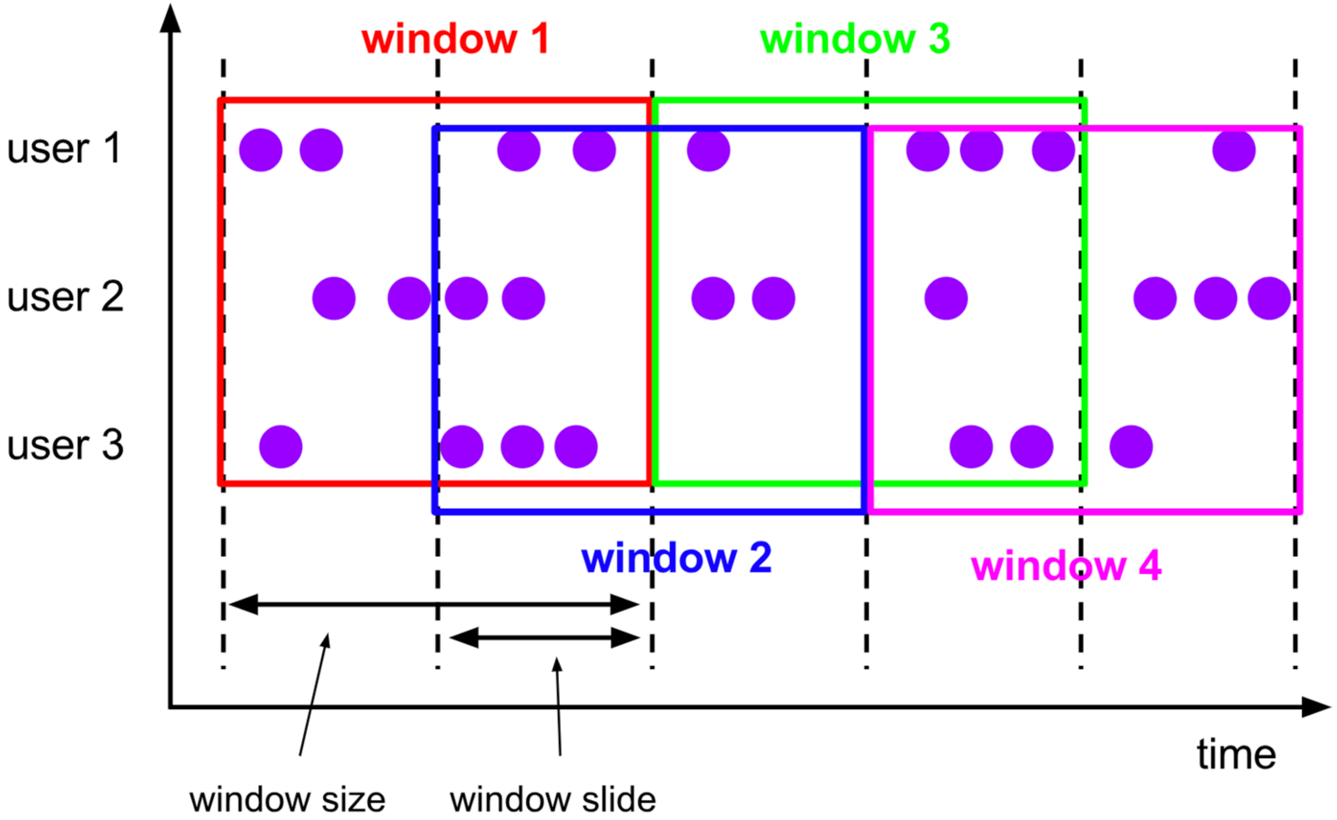
- Fixed-size, non-overlapping windows
- Each event belongs to exactly one window
- Useful for strict periodic aggregations
- Example: 5-minute intervals



Time-based windows: Tumbling, Sliding, Session

Sliding Windows:

- Fixed-size, potentially overlapping windows
- Events can belong to multiple windows
- Slide interval < window size
- Example: 10-minute window sliding every 5 minutes

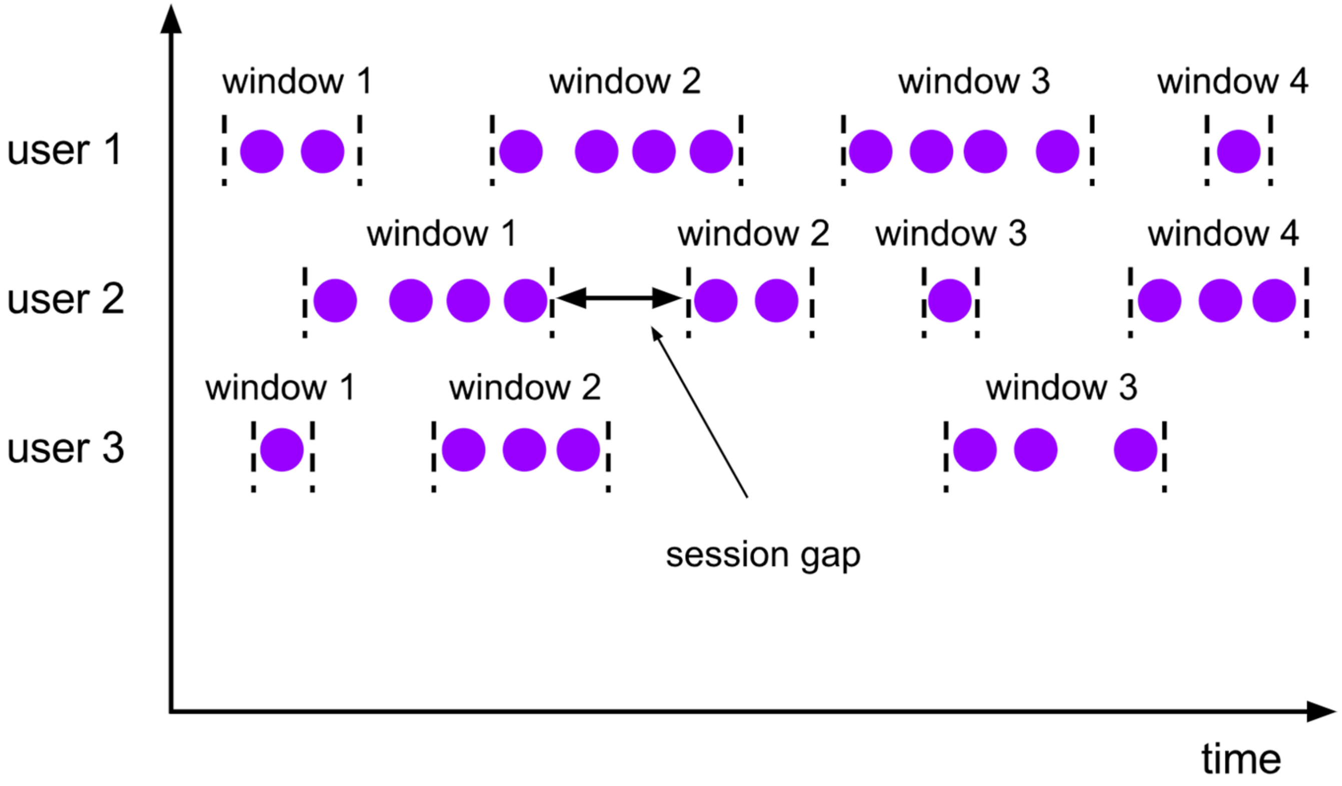


Time-based windows: Tumbling, Sliding, Session

Session Windows

Dynamic-sized windows based on inactivity

- Window closes after a period of no events
- Captures bursty user activity sessions
- Useful for analyzing user behavior patterns
- Example: Web sessions ending after 30s idle



Count Windows

Definition:

- A window that closes and fires computation after a specific number of events have arrived.
- Unlike time-based windows (Tumbling, Sliding, Session), Count Windows are **not tied to event time.**

Key Characteristics:

- Based purely on **element counts,** not timestamps.
- Guarantees aggregation after an **exact number of events.**
- Effective when events are irregular or when exact sample sizes are required.
- Simple to reason about — processing is triggered after the set threshold.

Count Windows

Advantages:

- Ensures deterministic processing per batch of N events.
- Works well for **batch-like processing within streams.**
- Handles irregular event rates without missing thresholds.

Limitations:

- Not aligned with time → cannot capture temporal trends.
- Late events after a window closes are not included.

Example:

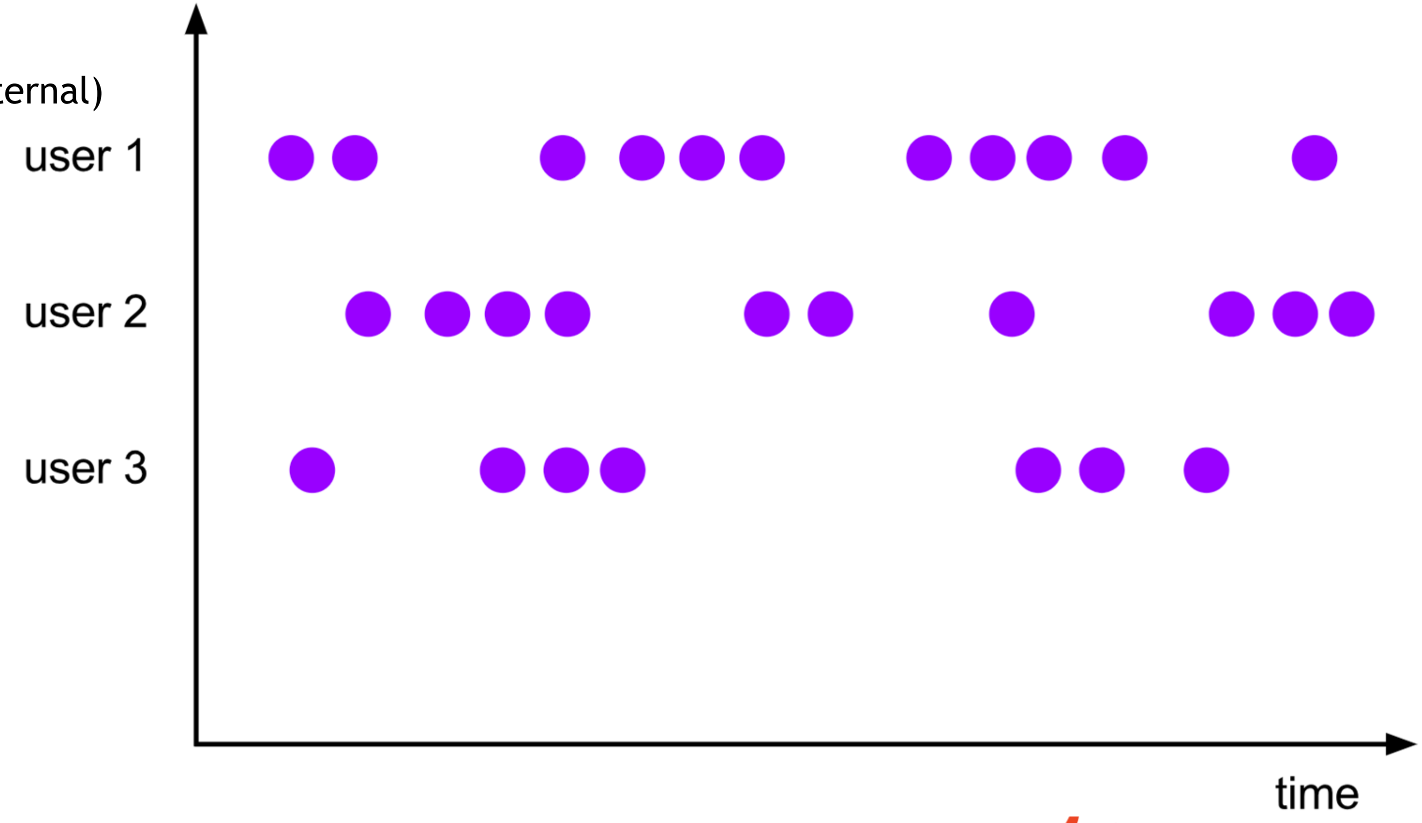
- With a count window size = 100:
  - Every **100 events**, an aggregation (sum, avg, etc.) is triggered.
  - If 235 events arrive → 2 full windows (100 + 100) and 1 partial window (35).

Global Windows

Definition: Entire stream as one logical window, closed only with explicit triggers.

Key Characteristics:

- No natural boundaries (time/count)
- Trigger-based (time, count, watermark, external)



Pros:

- Works for cumulative metrics

  (sum, average, total counts)

- Flexible trigger strategies

Cons:

- High memory use
- Careful trigger design required

Evictors

Evictors decide **which elements should be removed from a window's state** before or after the window function is applied.

- **Optional Component**
  Unlike triggers, evictors are not mandatory. They are useful when you need more control over **memory, performance, or data quality.**

- **How They Work**
  - **Before computation**: The evictor removes elements prior to running the window function (e.g., discarding outliers).
  - **After computation**: The evictor prunes the contents after results are computed (e.g., limiting state growth).

Use case: You may have a sliding window of 10 minutes, but you only want to keep the most recent 1000 events to reduce state size and latency.

Handling Late Data in Flink

1. Watermarks

- Mark event-time progress
- Signal that all events up to a certain time have likely arrived
- Determine when a window can be closed

2. Allowed Lateness

- Define how long windows wait for late events
- Example: allowedLateness = 5 min → events arriving within 5 minutes after watermark are still included

3. Very Late Data

- Events beyond allowed lateness are routed to a side output
- Enables logging, alerting, or special handling instead of dropping them

ANY SUGGESTIONS?


THANK YOU