Window & Functions in Flink

Preetdeep Kumar

30th Jan, 2020

https://www.linkedin.com/in/preetdeep-kumar/

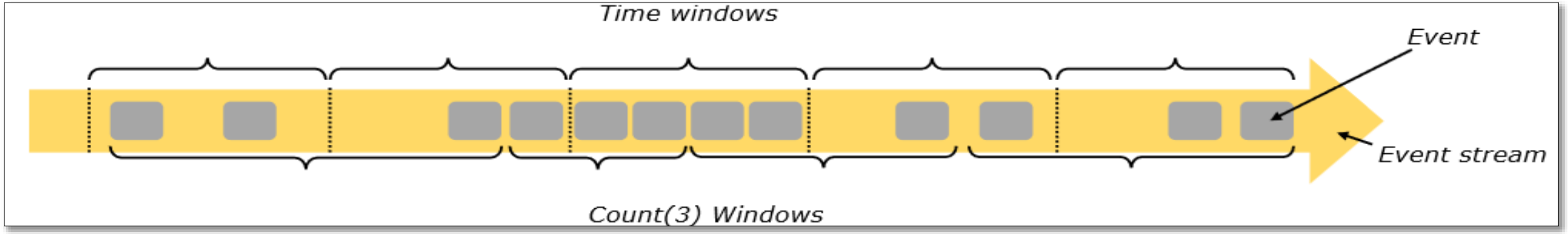
https://github.com/preetdeepkumar/flink-tutorials

https://www.meetup.com/Hyderabad-Apache-Flink-Meetup-Group/

Window in stream processing

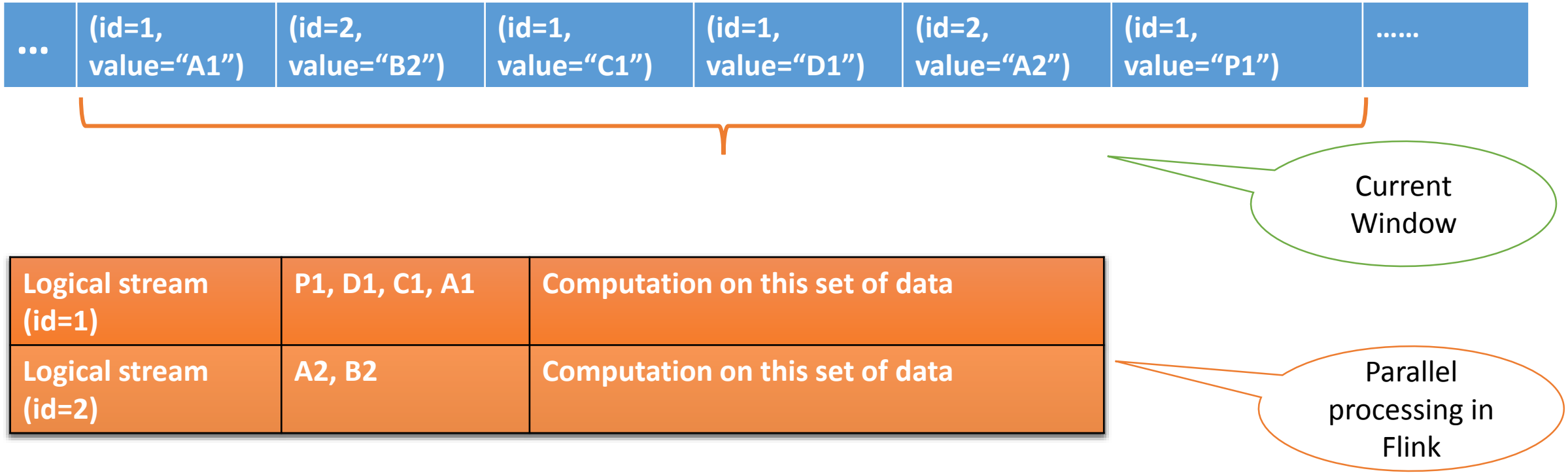
- A ‘data stream’ is never ending and continuously produces data thus it is not possible to compute a final value that can be returned.
- In most use cases, to get meaningful information two methods are preferred
  - Computation is done for a finite set over time (e.g. HTTP 401 errors per minute)
  - Computation is done as a rolling update (e.g. score board, trending topics)
- *A “Window” defines a finite set of elements on an unbounded stream over which we can apply computations*.
- This set can be based on time, element counts, a combination of counts and time, or some custom logic to assign elements to windows.
  - number of orders received every minute (fixed time)
  - average time to complete last 100 orders (fixed elements)

Window in stream processing – contd…



- Streaming framework vendors implements more than one variation of how a “Window” can be defined.
- For streams which are keyed, Flink will split your infinite stream into logical keyed streams
- Having a keyed stream will allow your windowed computation to be performed in parallel by multiple tasks, as each logical keyed stream can be processed independently from the rest.

Keyed vs Non-Keyed Windows



- All elements referring to the same key will be sent to the same parallel task.
- Very useful to speed up computation and integration with partition-based message source such as Kafka, AWS Kinesis

Window definitions in Flink

Tumbling window	Sliding window	Session window
Fixed size and do not overlap (no duplicate elements, each element is assigned to only one window)	Similar to tumbling but with a sliding value to start new window. Overlapping may occur (an element can be assigned to multiple window)	Do not overlap and does not have a fixed start and end time.
Time window of size of 5 minutes, will collect all elements arrived in a window and evaluate it after 5 minutes. A new window will be started every five minutes. Count window of 100 will collect 100 elements in a window and evaluate the window when the 100th element has been added.	Windows of size 10 minutes that slides by 5 minutes. With this you get every 5 minutes a window that contains the events that arrived during the last 10 minute	Starts when an element arrives and closes when an element hasn't arrived for a certain period of time. When this period expires, the current session closes and subsequent elements are assigned to a new session window.

Window Functions

- After defining the window, we need to specify the business logic required on each of these windows.
- This is the responsibility of the *window function*, which is used to process the elements of each window once the system determines that a window is ready for processing
- Two categories of window functions: Incremental and Non-Incremental computation
- Incremental (*elements are processed as they arrive in window)*
  - ReduceFunction
  - AggregateFunction
  - FoldFunction
- Non-Incremental *(all elements are buffered in-memory before processing)*
  - ProcessWindowFunction

Demo

References

https://flink.apache.org/news/2015/12/04/Introducing-windows.html

https://ci.apache.org/projects/flink/flink-docs-release-1.9/concepts/programming-model.html#windows

https://ci.apache.org/projects/flink/flink-docs-release-1.9/dev/stream/operators/windows.html

Backup

Window Lifecycle in Flink

- A window is created as soon as the first element that should belong to this window arrives, and the window is completely removed when the time (event or processing time) passes its end timestamp plus the *user-specified allowed lateness*
- In addition, each window will have a “Trigger” and a “Function” (ProcessWindowFunction, ReduceFunction, AggregateFunction or FoldFunction) attached to it.
- The “function” will contain the computation (business logic) to be applied to the contents of the window, while the Trigger specifies the conditions under which the window is considered ready for the function to be applied.

How does the stream processor interpret time

- **Event Time** is the time when an event was created. It is usually described by a timestamp in the events.
- **Ingestion time** is the time when an event enters the Flink dataflow at the source operator.
- **Processing Time** is the local time at each operator that performs a time-based operation.