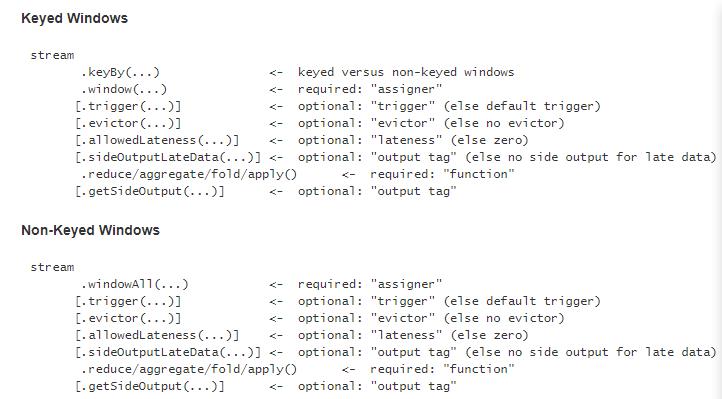
1,简述

窗口是无限流处理的核心，窗口可以把无限流分割成有限流进行计算。窗口可以分为有key和无key的窗口.



2.窗口的周期

窗口被创建不久，数据就会到达这个窗口，基于这个窗口的数据和允许延迟的的数据，消费完后窗口被完全移除。 Flink保证只删除基于时间的窗口，而不删除其他类型的窗口，例如全局窗口。例如，我们创建一个5分钟不重叠的窗口并且延迟1分钟，窗口的时间间隔12:00—12：05 ，当水印的时间戳是12：06时窗口将要被移除。

每个窗口都有一个触发器和应用到这个窗口的函数。这个函数是处理这个窗口的内容，而触发器在指定的条件下进行计算，触发器的策略可能是在窗口内的元素的个数大于4或者当水印通过窗口末端。触发器还可以决定在窗口创建和删除之间的任何时间清除其内容。清除只涉及窗口中的元素，而不是窗口元数据。您可以指定一个evictor(逐出器)，该evictor将能够在触发触发器之后、应用函数之前和/或之后从窗口中移除元素。

3.key vs no-key 窗口

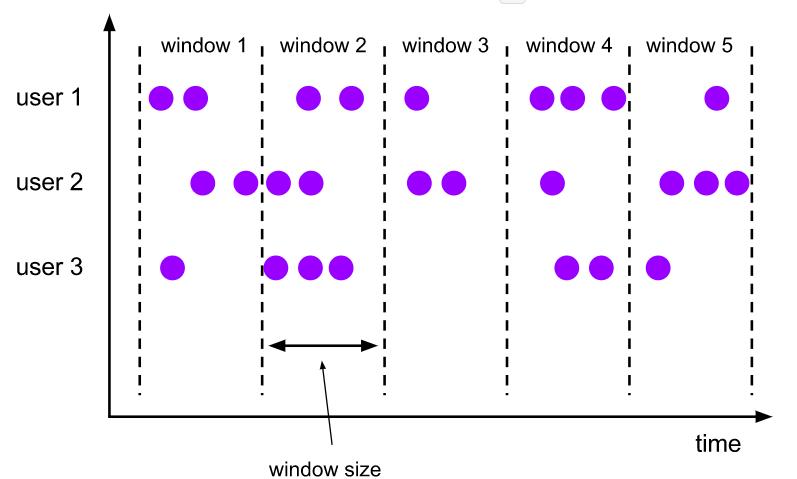
在使用窗口的第一件事情就是制定键值。如果制定key，窗口内的数据按照key划分多个流并提高并行性，所有相同的key被划分同一个任务中。如果不制定key，所有的数据就只有一个任务，并行性为1。

4. 窗口分配器

在你指定好是否是有key以后，下一步就要指定窗口分配器，窗口分配器定义元素怎样分配给窗口。窗口分配器最通用的窗口: tumbling windows, sliding windows, session windows and global windows,你也可以通继承WindowAssigner 类来实现自定义窗口分配器,窗口分配器基于处理时间和事件事件分配元素。窗口是基于开始时间戳和结束时间戳来决定窗口的大小。

1. tumbling windows

tumbling windows是固定大小并且不会有重叠（两个窗口之间没有重叠）。



代码:

**val** input**:** **DataStream[T]** **=** **...**

*// tumbling event-time windows*

input

**.**keyBy**(<**key selector**>)**

**.**window**(TumblingEventTimeWindows.**of**(Time.**seconds**(**5**)))**

**.<**windowed transformation**>(<**window function**>)**

*// tumbling processing-time windows*

input

**.**keyBy**(<**key selector**>)**

**.**window**(TumblingProcessingTimeWindows.**of**(Time.**seconds**(**5**)))**

**.<**windowed transformation**>(<**window function**>)**

*// daily tumbling event-time windows offset by -8 hours.*

input

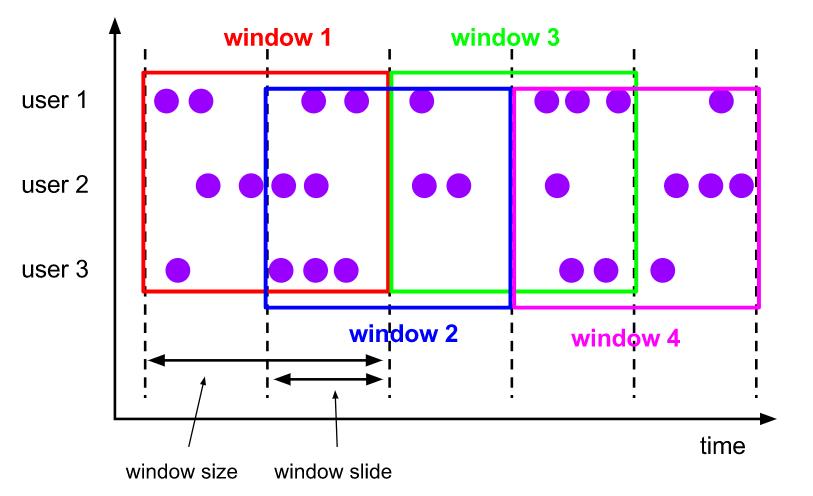
**.**keyBy**(<**key selector**>)**

**.**window**(TumblingEventTimeWindows.**of**(Time.**days**(**1**),** **Time.**hours**(-**8**)))**

**.<**windowed transformation**>(<**window function**>)**

### Sliding Windows

### Sliding windows分配一个固定大小的窗口，第一个参数是窗口的大小，第二个是窗口开始执行计算的时间，如果两个参数相等，就和tumbling Windows一样，如果两个值不相等，元素有可能重复。



代码:

**val** input**:** **DataStream[T]** **=** **...**

*// sliding event-time windows*

input

**.**keyBy**(<**key selector**>)**

**.**window**(SlidingEventTimeWindows.**of**(Time.**seconds**(**10**),** **Time.**seconds**(**5**)))**

**.<**windowed transformation**>(<**window function**>)**

*// sliding processing-time windows*

input

**.**keyBy**(<**key selector**>)**

**.**window**(SlidingProcessingTimeWindows.**of**(Time.**seconds**(**10**),** **Time.**seconds**(**5**)))**

**.<**windowed transformation**>(<**window function**>)**

*// sliding processing-time windows offset by -8 hours*

input

**.**keyBy**(<**key selector**>)**

**.**window**(SlidingProcessingTimeWindows.**of**(Time.**hours**(**12**),** **Time.**hours**(**1**),** **Time.**hours**(-**8**)))**

**.<**windowed transformation**>(<**window function**>)**