toc: true title: 《从0到1学习Flink》—— Flink 中几种 Time 详解 date: 2018-12-11 tags:

- Flink
- 大数据
- 流式计算

前言

Flink 在流程序中支持不同的 **Time** 概念，就比如有 Processing Time、Event Time 和 Ingestion Time。

下面我们一起来看看这几个 Time：

Processing Time

Processing Time 是指事件被处理时机器的系统时间。

当流程序在 Processing Time 上运行时，所有基于时间的操作(如时间窗口)将使用当时机器的系统时间。每小时 Processing Time 窗口将包括在系统时钟指示整个小时之间到达特定操作的所有事件。

例如，如果应用程序在上午 9:15 开始运行，则第一个每小时 Processing Time 窗口将包括在上午 9:15 到上午 10:00 之间处理的事件，下一个窗口将包括在上午 10:00 到 11:00 之间处理的事件。

Processing Time 是最简单的 "Time" 概念，不需要流和机器之间的协调，它提供了最好的性能和最低的延迟。但是，在分布式和异步的环境下，Processing Time 不能提供确定性，因为它容易受到事件到达系统的速度（例如从消息队列）、事件在系统内操作流动的速度以及中断的影响。

Event Time

Event Time 是事件发生的时间，一般就是数据本身携带的时间。这个时间通常是在事件到达 Flink 之前就确定的，并且可以从每个事件中获取到事件时间戳。在 Event Time 中，时间取决于数据，而跟其他没什么关系。Event Time 程序必须指定如何生成 Event Time 水印，这是表示 Event Time 进度的机制。

完美的说，无论事件什么时候到达或者其怎么排序，最后处理 Event Time 将产生完全一致和确定的结果。但是，除非事件按照已知顺序（按照事件的时间）到达，否则处理 Event Time 时将会因为要等待一些无序事件而产生一些延迟。由于只能等待一段有限的时间，因此就难以保证处理 Event Time 将产生完全一致和确定的结果。

假设所有数据都已到达， Event Time 操作将按照预期运行，即使在处理无序事件、延迟事件、重新处理历史数据时也会产生正确且一致的结果。 例如，每小时事件时间窗口将包含带有落入该小时的事件时间戳的所有记录，无论它们到达的顺序如何。

请注意，有时当 Event Time 程序实时处理实时数据时，它们将使用一些 Processing Time 操作，以确保它们及时进行。

Ingestion Time

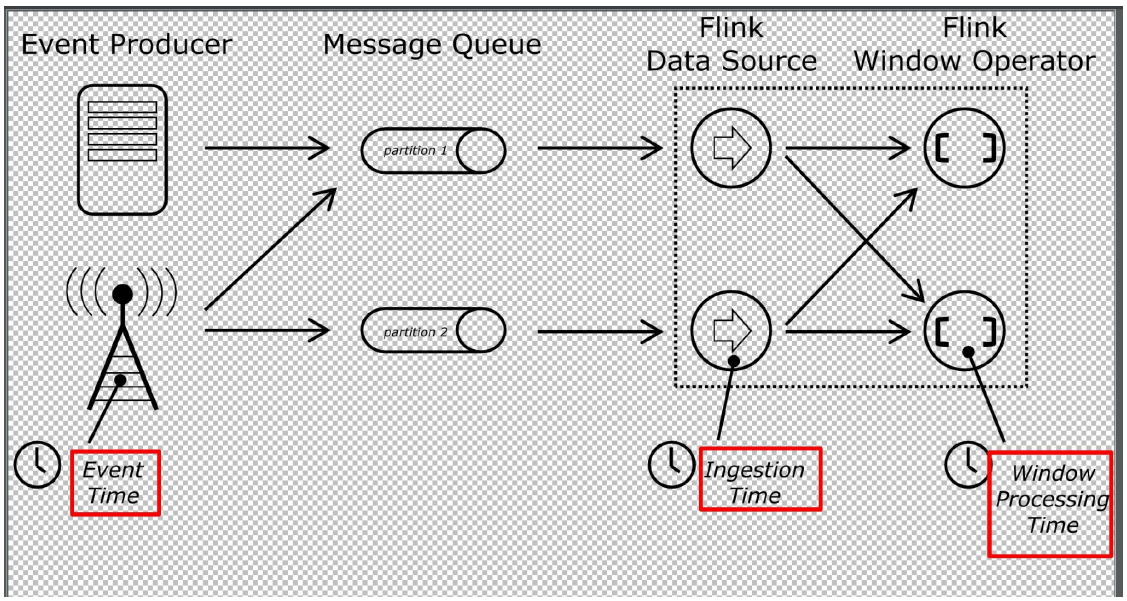
Ingestion Time 是事件进入 Flink 的时间。 在源操作处，每个事件将源的当前时间作为时间戳，并且基于时间的操作（如时间窗口）会利用这个时间戳。

Ingestion Time 在概念上位于 Event Time 和 Processing Time 之间。 与 Processing Time 相比，它稍微贵一些，但结果更可预测。因为 Ingestion Time 使用稳定的时间戳（在源处分配一次），所以对事件的不同窗口操作将引用相同的时间戳，而在 Processing Time 中，每个窗口操作符可以将事件分配给不同的窗口（基于机器系统时间和到达延迟）。

与 Event Time 相比，Ingestion Time 程序无法处理任何无序事件或延迟数据，但程序不必指定如何生成水印。

在 Flink 中，，Ingestion Time 与 Event Time 非常相似，但 Ingestion Time 具有自动分配时间戳和自动生成水印功能。

说了这么多概念比较干涩，下面直接看图：



设定时间特性

Flink DataStream 程序的第一部分通常是设置基本时间特性。 该设置定义了数据流源的行为方式（例如：它们是否将分配时间戳），以及像 `KeyedStream.timeWindow(Time.seconds(30))` 这样的窗口操作应该使用上面哪种时间概念。

以下示例显示了一个 Flink 程序，该程序在每小时时间窗口中聚合事件。

```
final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

env.setStreamTimeCharacteristic(TimeCharacteristic.ProcessingTime);

// 其他
//
env.setStreamTimeCharacteristic(TimeCharacteristic.IngestionTime);
// env.setStreamTimeCharacteristic(TimeCharacteristic.EventTime);

DataStream<MyEvent> stream = env.addSource(new
FlinkKafkaConsumer09<MyEvent>(topic, schema, props));

stream
    .keyBy( (event) -> event.getUser() )
    .timeWindow(Time.hours(1))
    .reduce( (a, b) -> a.add(b) )
    .addSink(...);
```

Event Time 和 Watermarks

注意：Flink 实现了数据流模型中的许多技术。有关 Event Time 和 Watermarks 的详细介绍，请查看以下文章：

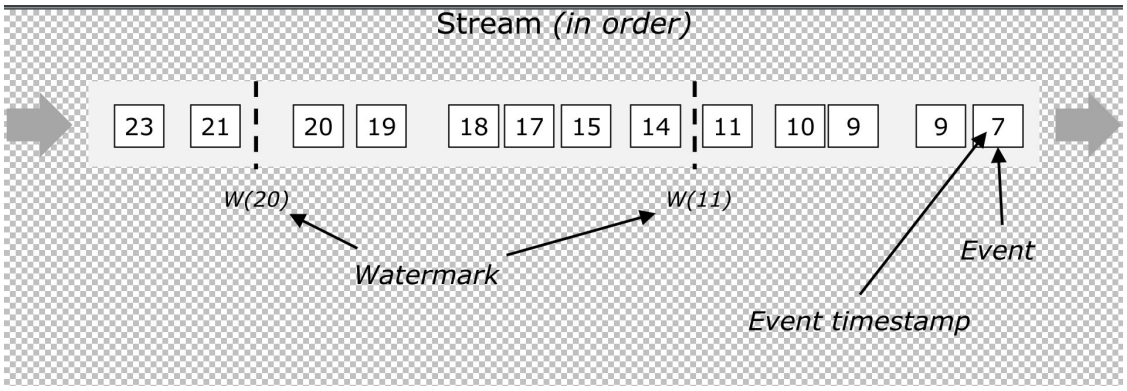
- https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101
- https://research.google.com/pubs/archive/43864.pdf

支持 Event Time 的流处理器需要一种方法来衡量 Event Time 的进度。 例如，当 Event Time 超过一小时结束时，需要通知构建每小时窗口的窗口操作符，以便操作员可以关闭正在进行的窗口。

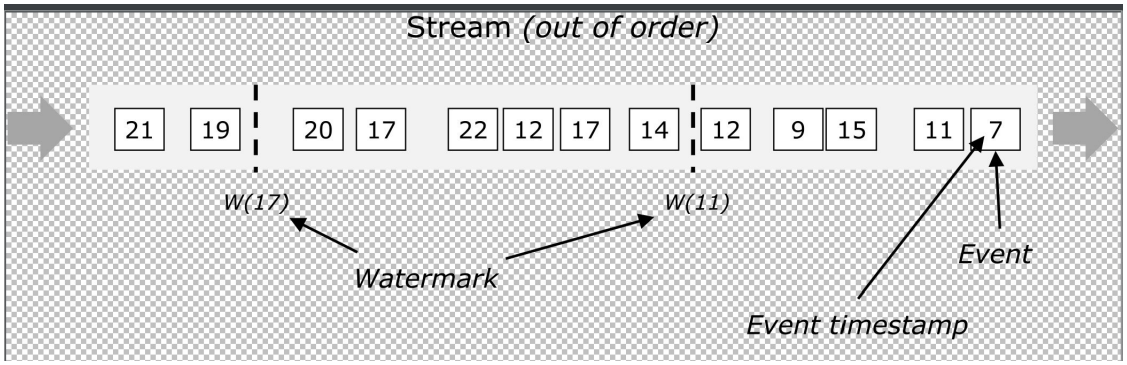
Event Time 可以独立于 Processing Time 进行。 例如，在一个程序中，操作员的当前 Event Time 可能略微落后于 Processing Time （考虑到接收事件的延迟），而两者都以相同的速度进行。另一方面，另一个流程序可能只需要几秒钟的时间就可以处理完 Kafka Topic 中数周的 Event Time 数据。

Flink 中用于衡量 Event Time 进度的机制是 Watermarks。 Watermarks 作为数据流的一部分流动并带有时间戳 t。 Watermark（t）声明 Event Time 已到达该流中的时间 t，这意味着流中不应再有具有时间戳 t'<= t 的元素（即时间戳大于或等于水印的事件）

下图显示了带有(逻辑)时间戳和内联水印的事件流。在本例中，事件是按顺序排列的(相对于它们的时间戳)，这意味着水印只是流中的周期性标记。



Watermark 对于无序流是至关重要的，如下所示，其中事件不按时间戳排序。通常，Watermark 是一种声明，通过流中的该点，到达某个时间戳的所有事件都应该到达。一旦水印到达操作员，操作员就可以将其内部事件时间提前到水印的值。



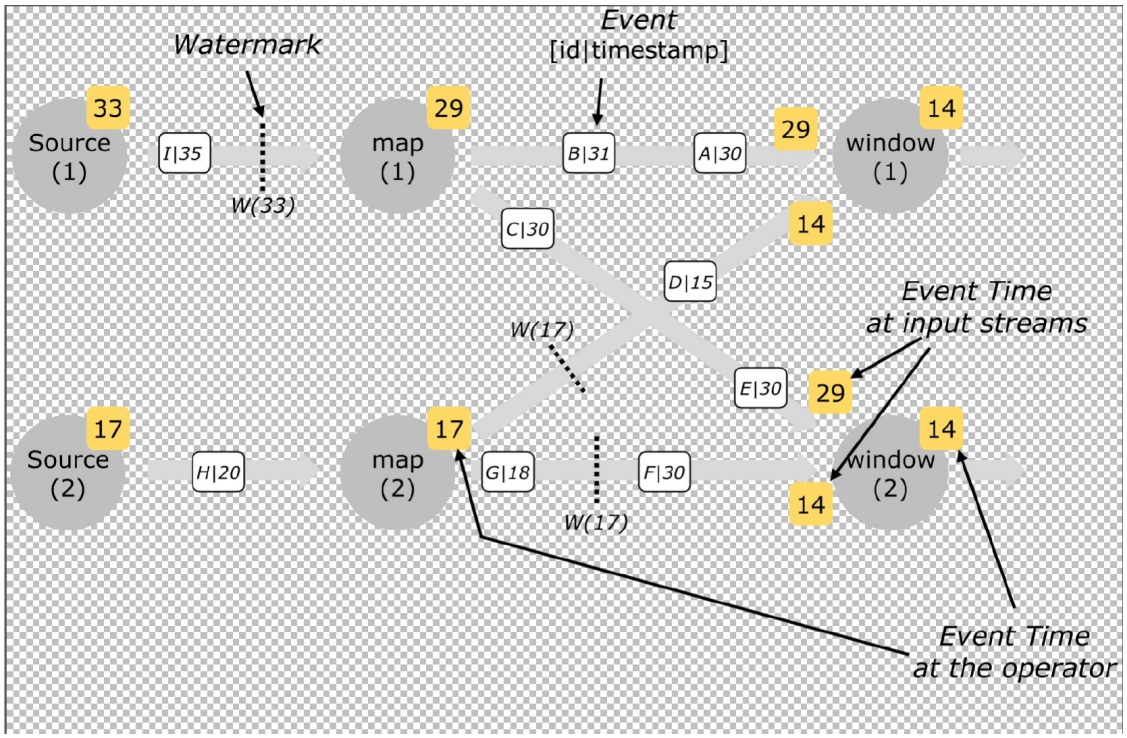
平行流中的水印

水印是在源函数处生成的，或直接在源函数之后生成的。源函数的每个并行子任务通常独立生成其水印。这些水印定义了特定并行源处的事件时间。

当水印通过流程序时，它们会提前到达操作人员处的事件时间。当一个操作符提前它的事件时间时，它为它的后续操作符在下游生成一个新的水印。

一些操作员消耗多个输入流; 例如，一个 union，或者跟随 keyBy（...）或 partition（...）函数的运算符。 这样的操作员当前事件时间是其输入流的事件时间的最小值。 由于其输入流更新其事件时间，因此操作员也是如此。

下图显示了流经并行流的事件和水印的示例，以及跟踪事件时间的运算符。



参考

https://github.com/zhisheng17/flink/blob/feature%2Fzhisheng_release_1.6/docs/dev/event_time.md

关注我

转载请务必注明原创地址为：http://www.54tianzhisheng.cn/2018/12/11/Flink-time/

另外我自己整理了些 Flink 的学习资料，目前已经全部放到微信公众号了。你可以加我的微信：zhisheng_tian，然后回复关键字：Flink 即可无条件获取到。

Github 代码仓库

https://github.com/zhisheng17/flink-learning/

以后这个项目的所有代码都将放在这个仓库里，包含了自己学习 flink 的一些 demo 和博客

相关文章

1、《从0到1学习Flink》—— Apache Flink 介绍

2、《从0到1学习Flink》—— Mac 上搭建 Flink 1.6.0 环境并构建运行简单程序入门

3、《从0到1学习Flink》—— Flink 配置文件详解

4、《从0到1学习Flink》—— Data Source 介绍

5、《从0到1学习Flink》—— 如何自定义 Data Source ？

6、《从0到1学习Flink》—— Data Sink 介绍

7、《从0到1学习Flink》—— 如何自定义 Data Sink？

8、《从0到1学习Flink》—— Flink Data transformation(转换)

9、《从0到1学习Flink》—— 介绍Flink中的Stream Windows

10、《从0到1学习Flink》—— Flink 中的几种 Time 详解

11、《从0到1学习Flink》—— Flink 写入数据到 ElasticSearch