Apache Beam: 流处理与批处理的统一编程框架

何沛
阿里巴巴-中间件-实时计算
Apache Beam committer

Based on slides by Tyler Akidau & Frances Perry.

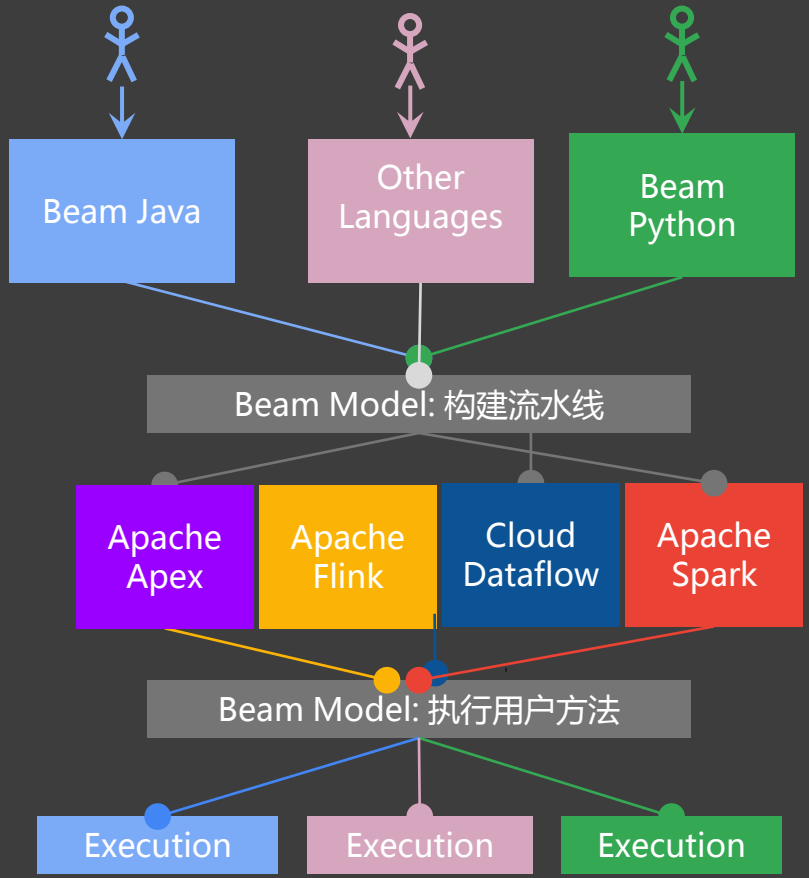
Apache Beam

定义了流处理与批处理*统一的编程模型，*

它生成的流水线具有**很好的***可移植性，*

*它在执行层面定义了**高效的执行协议。***

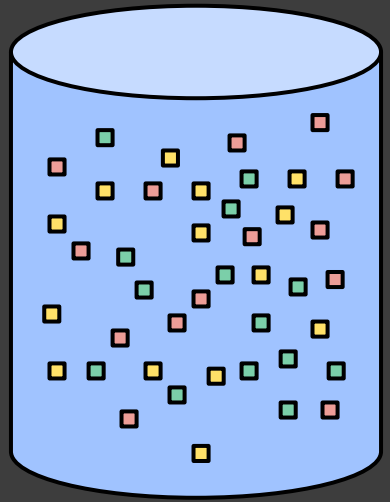
1. *Apache Beam 编程模型*
   - What Where When How
2. *SDK 构建流水线*
   - *Java*
   - *Python*
3. *执行引擎 Runners*
   - *Apex*
   - *Spark*
   - *Flink*
   - *Google Cloud Dataflow*



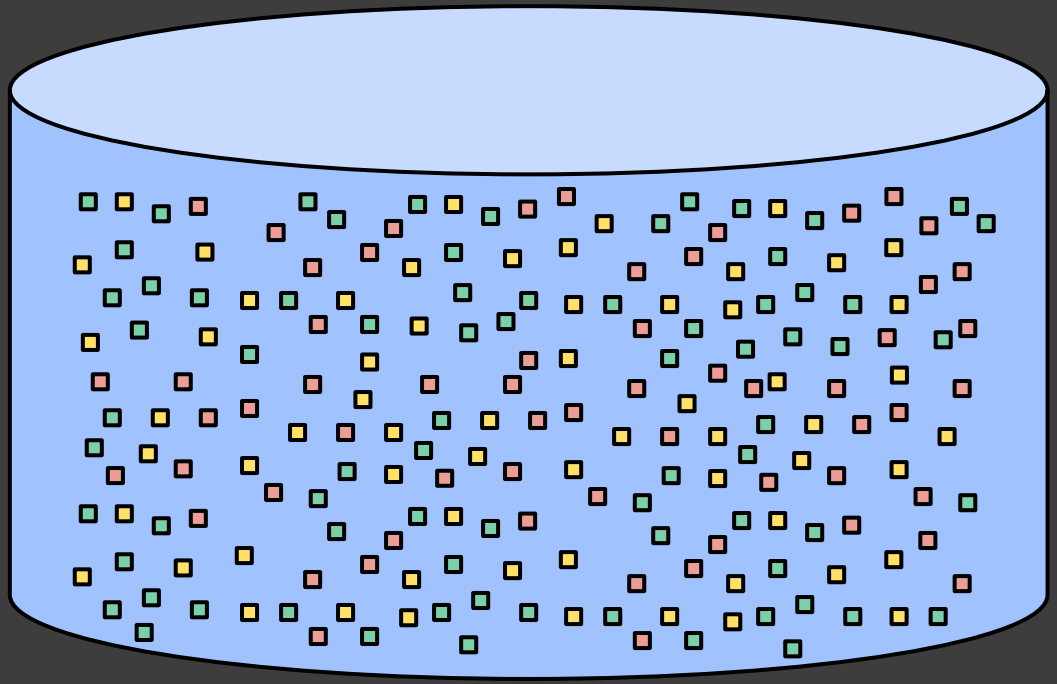
流处理与批处理统一编程模型

https://commons.wikimedia.org/wiki/File:Globe_centered_in_the_Atlantic_Ocean_(green_and_grey_globe_scheme).svg

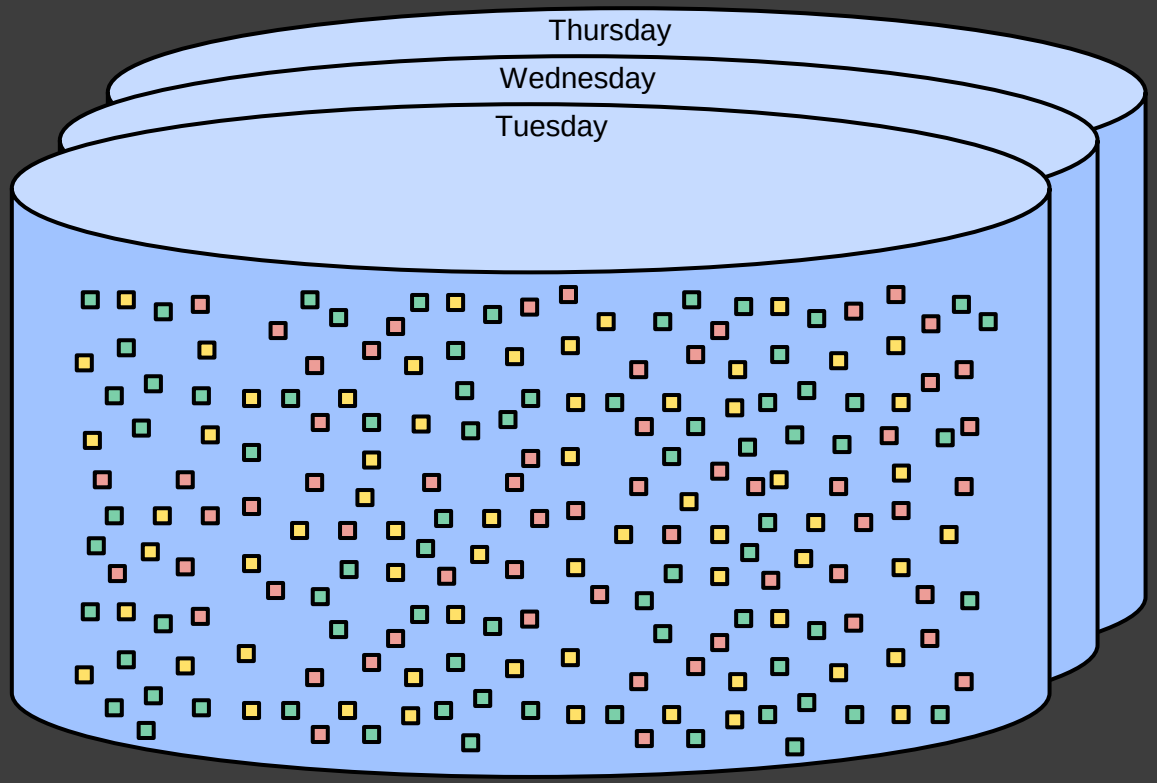
数据...



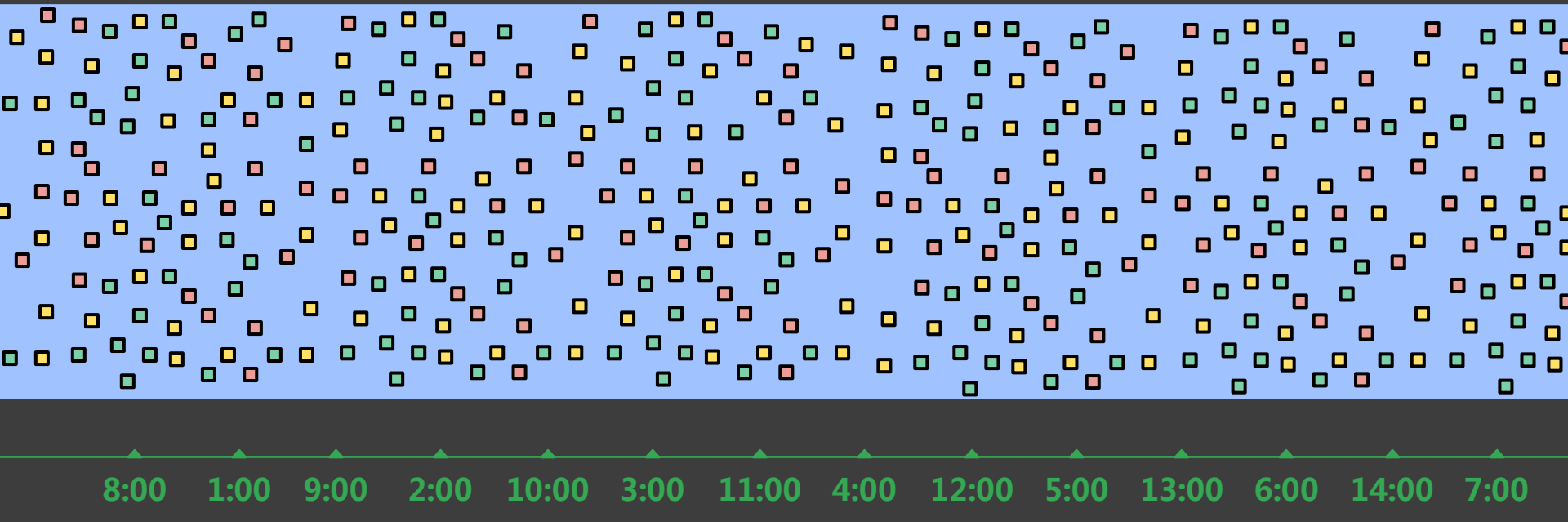
...很多数据...



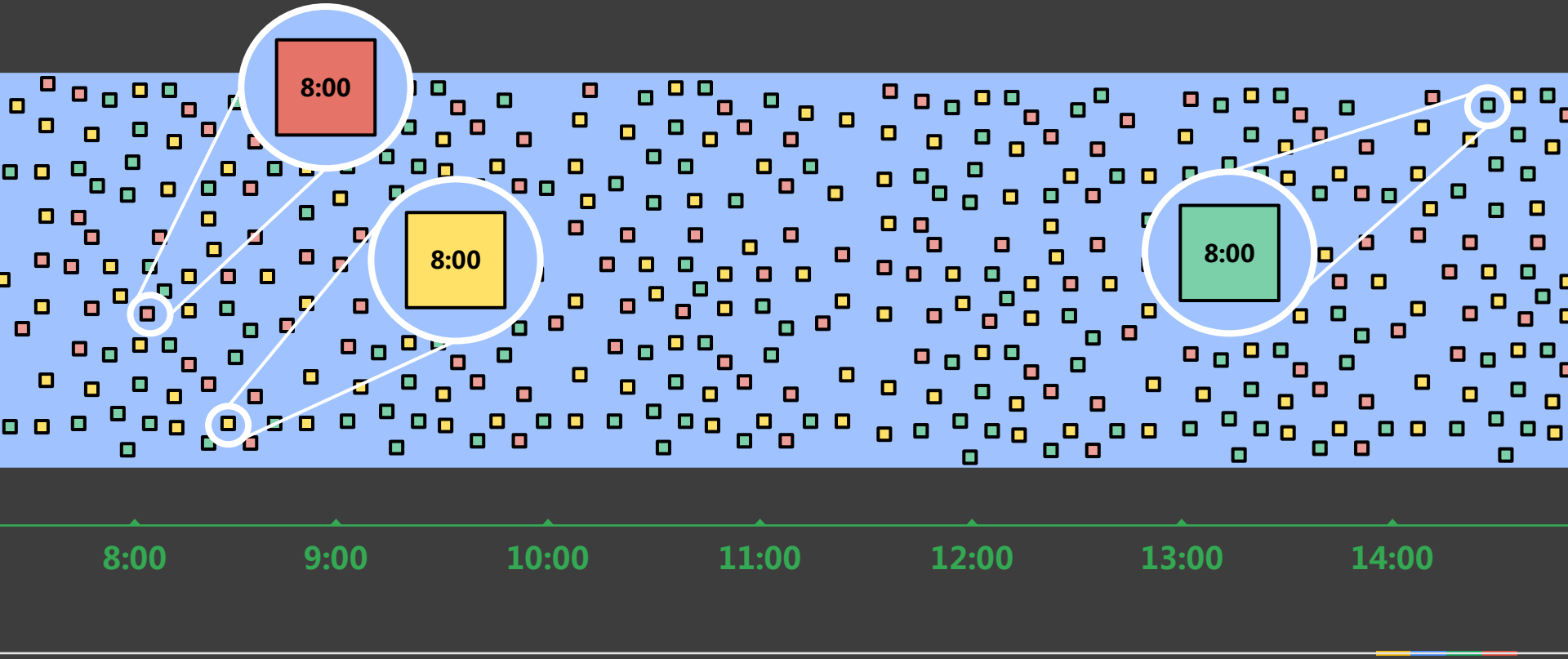
...越来越多的数据...



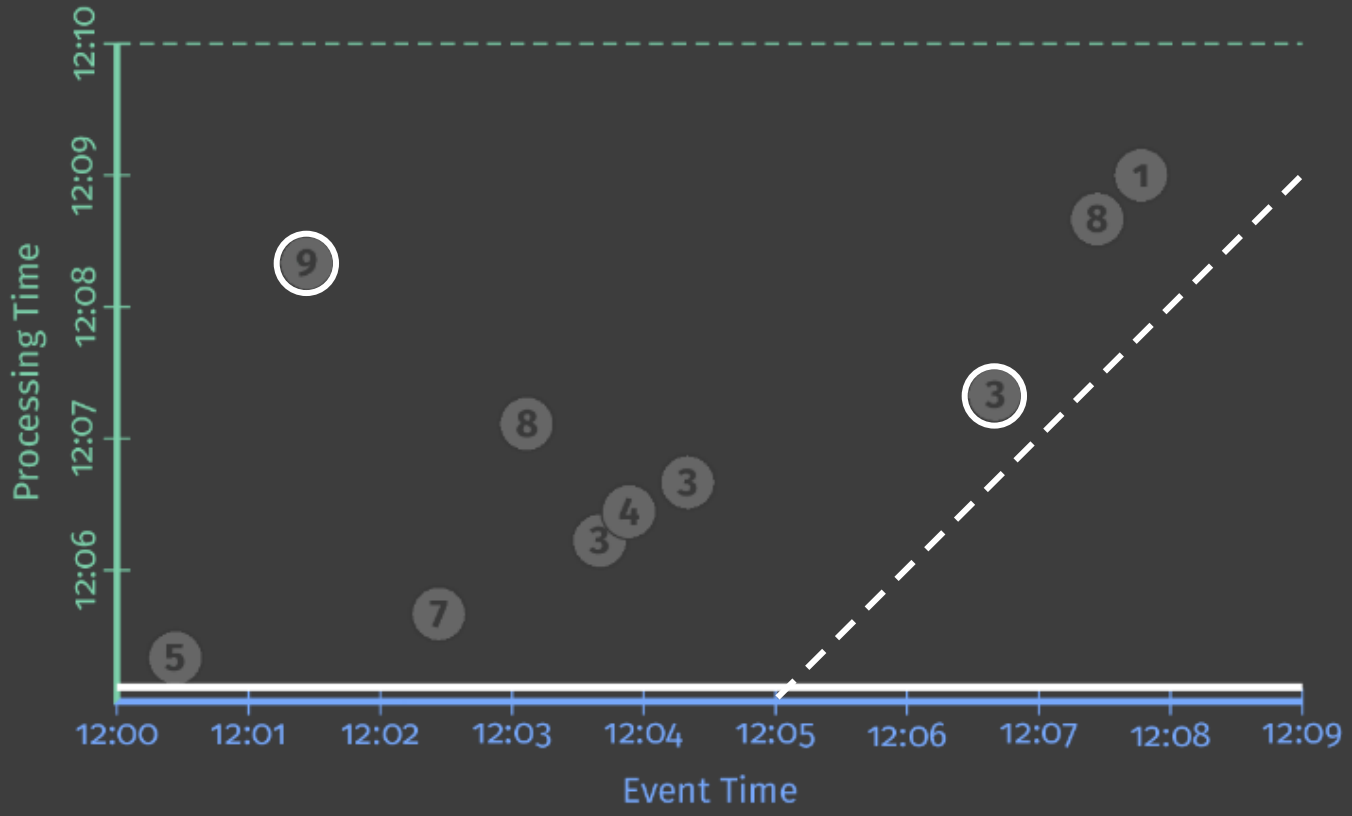
… 无边界的流数据…



...无序、有延迟的流数据.



事件时间 (Event time) 与 处理时间 (Processing Time)



What 业务逻辑是什么? (ParDo, Combine)

Where 在事件时间轴上对应哪里? (Windowing)

When 何时输出结果？(Triggering)

How 如何更新结果？(Accumulating)

What: 求和 (Sum)

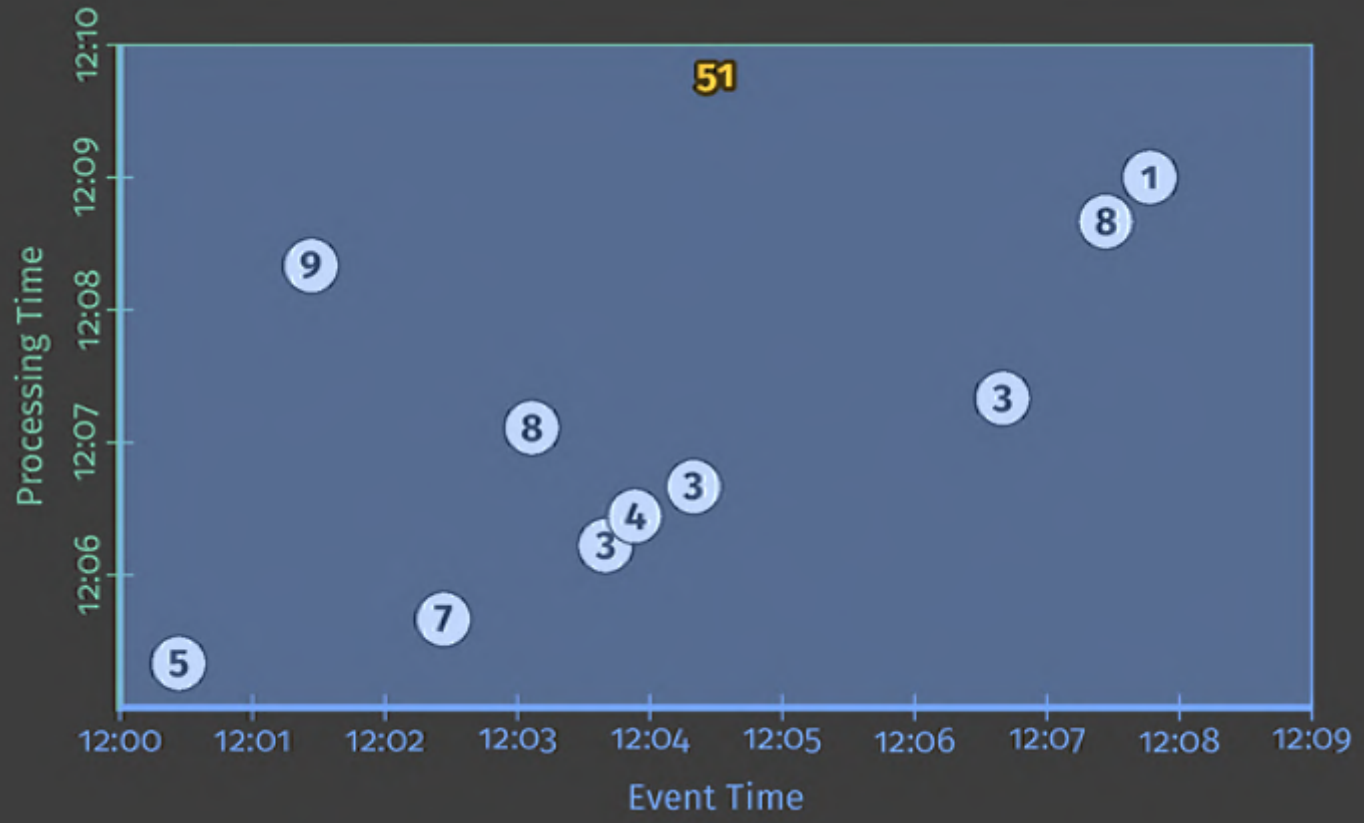
```
// Collection of raw log lines
PCollection<String> raw = IO.read(...);

// Element-wise transformation into team/score pairs
PCollection<KV<String, Integer>> input =
     raw.apply(ParDo.of(new ParseFn());

// Composite transformation containing an aggregation
PCollection<KV<String, Integer>> scores =
     input.apply(Sum.integersPerKey());
```

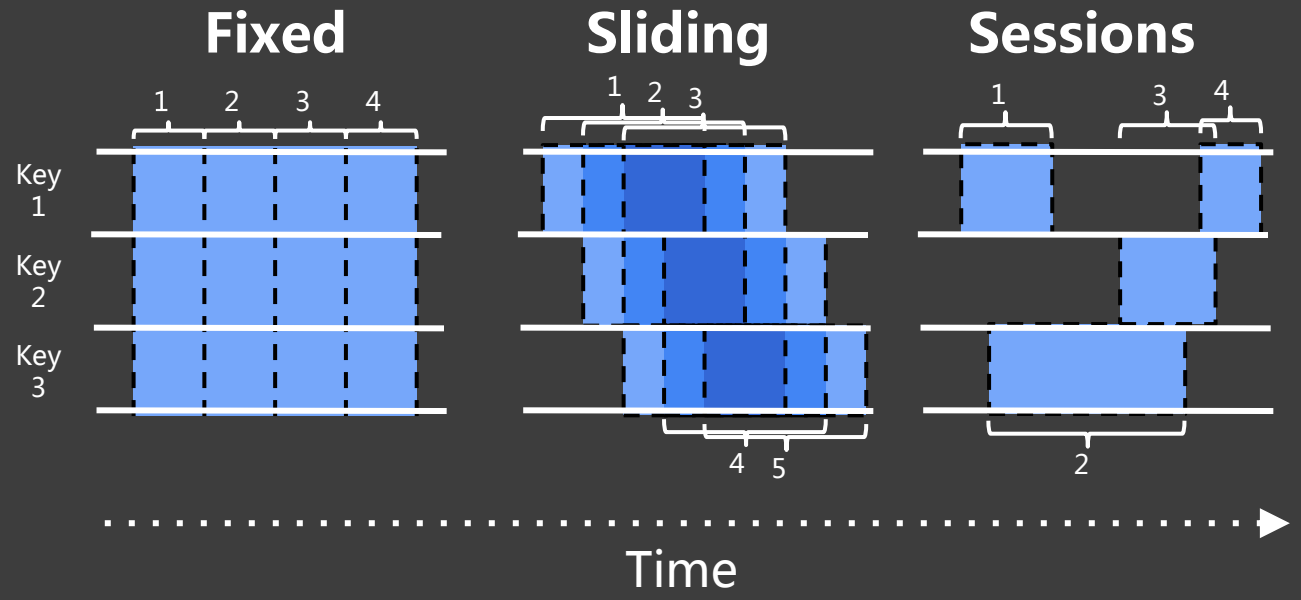
*All code snippets are pseudo-java -- details shortened or elided for clarity.

What: 求和 (Sum)


51
9
8
3
8
3
4
3
7
5
1
Processing Time
12:10
12:09
12:08
12:07
12:06
12:00
12:01
12:02
12:03
12:04
12:05
12:06
12:07
12:08
12:09
Event Time

Where: 在事件时间轴上对应哪里?

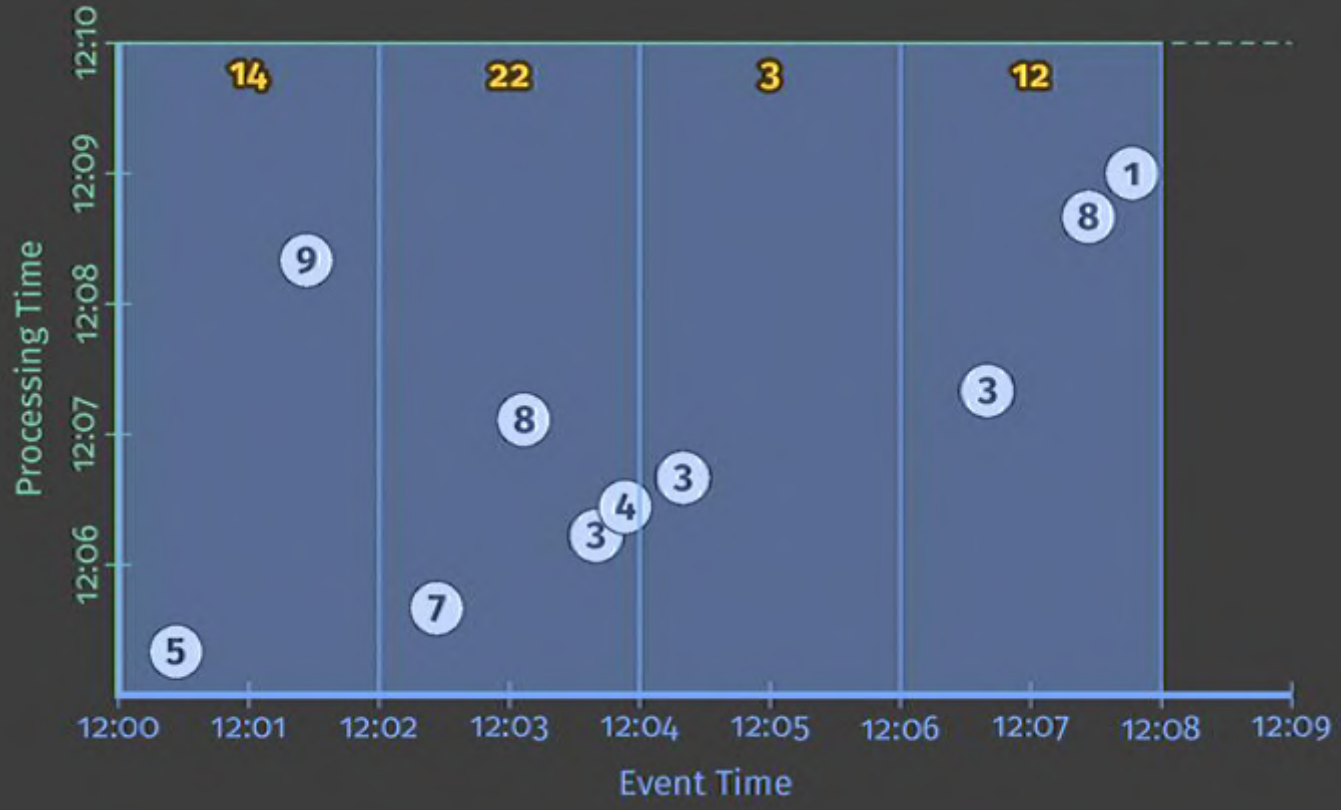
Windowing 可以将流数据划分为有边界的数据集



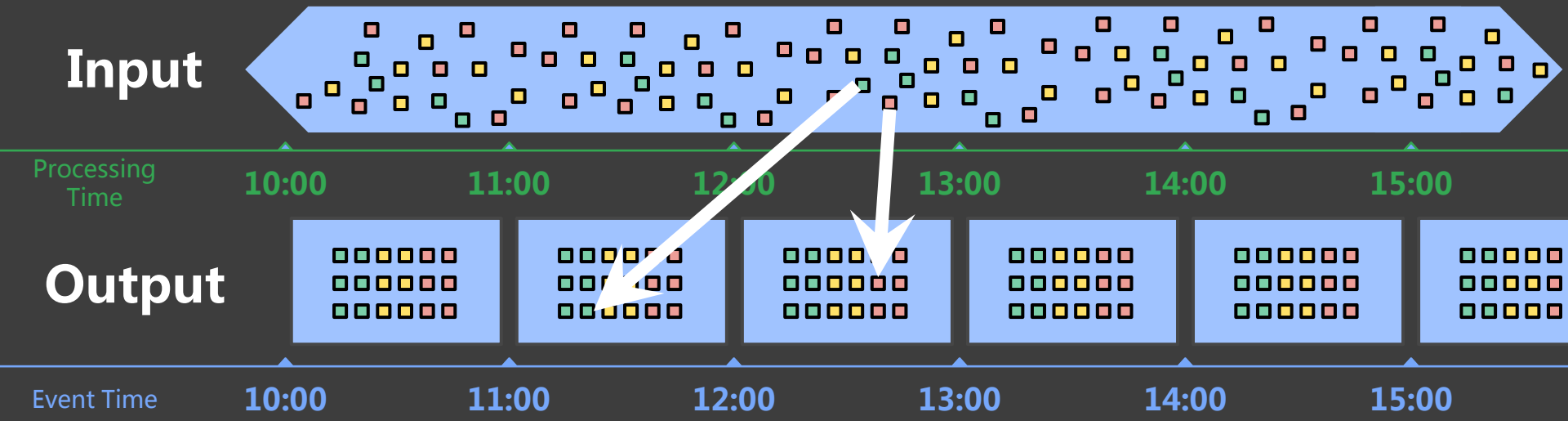
Where: 两分钟 固定窗口

```
PCollection<KV<String, Integer>> scores = input
    .apply(Window
       .into(FixedWindows.of(Duration.standardMinutes(2))))
    .apply(Sum.integersPerKey());
```

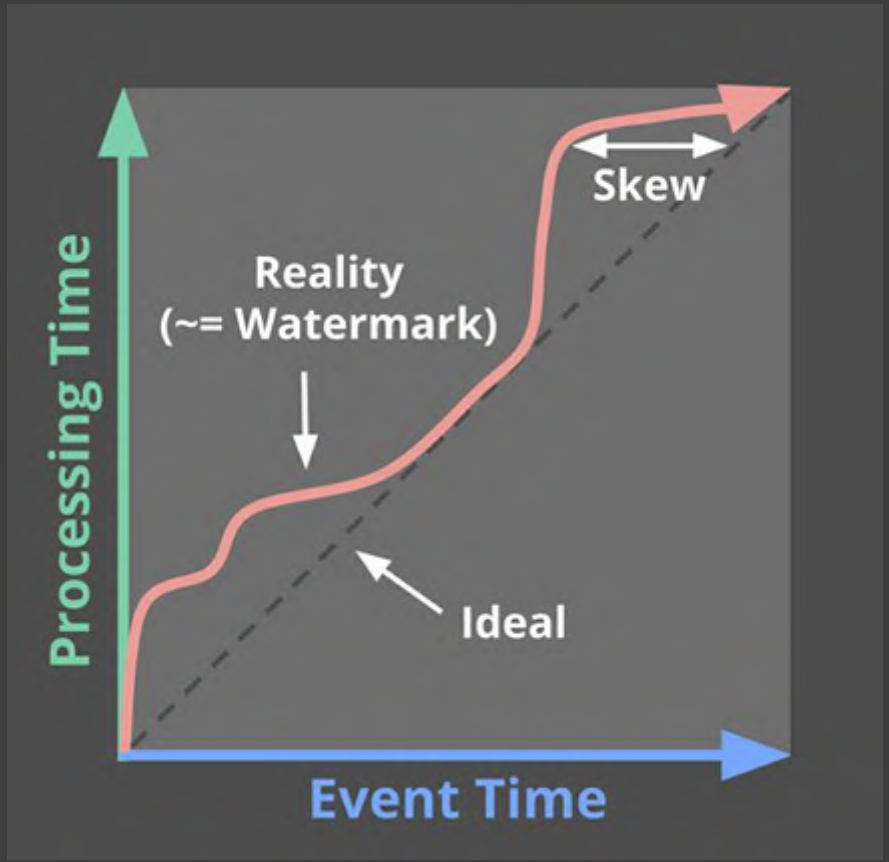
Where: 两分钟 固定窗口



流数据处理: 引入Watermark的概念



When: 何时输出计算结果?



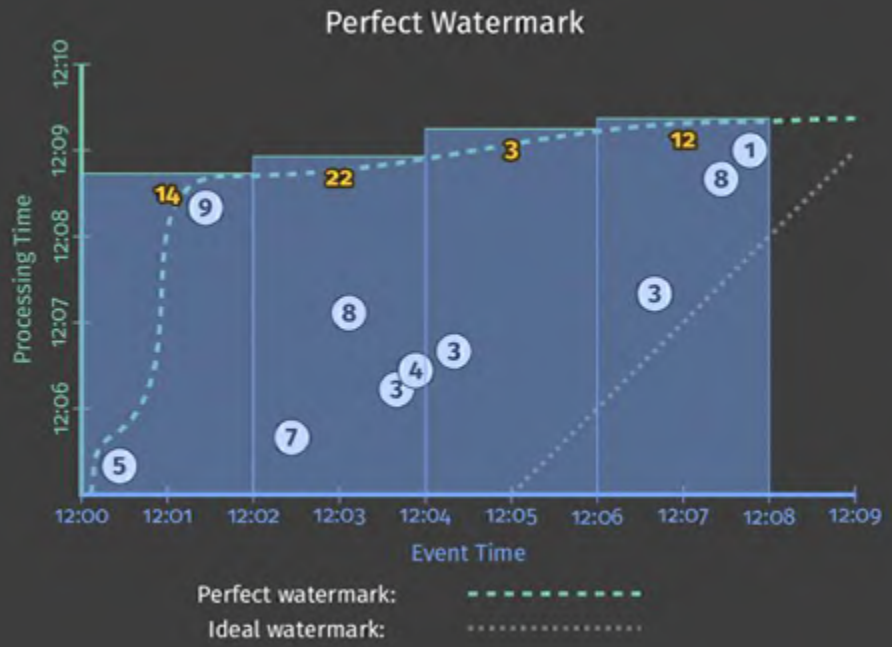
Triggers控制结果输出

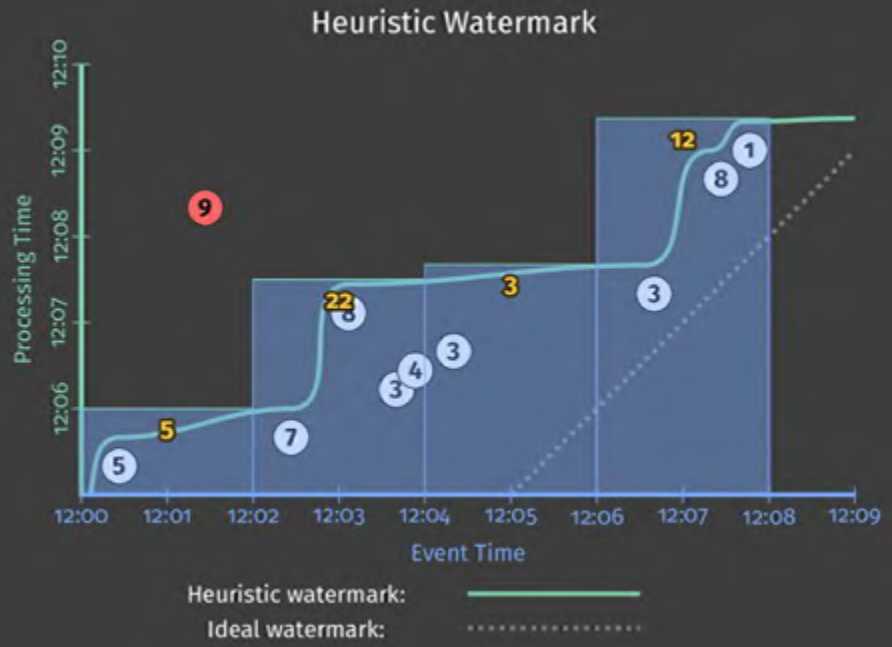
```
triggering(AtWatermark())
```

When: 按时输出（Triggering at the Watermark）

```
PCollection<KV<String, Integer>> scores = input
    .apply(Window
        .into(FixedWindows.of(Duration.standardMinutes(2))
        .triggering(AtWatermark()))
    .apply(Sum.integersPerKey());
```

When: 按时输出（Triggering at the Watermark）

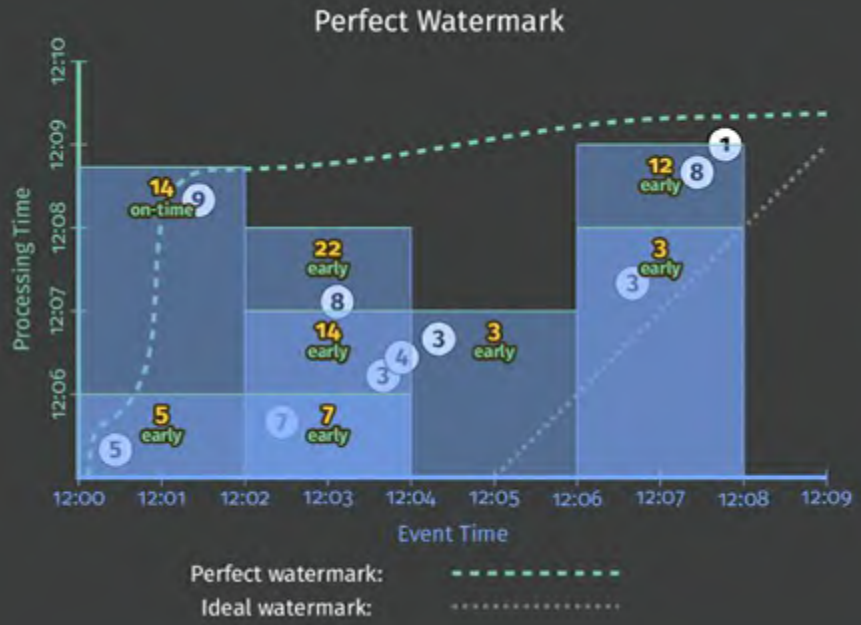


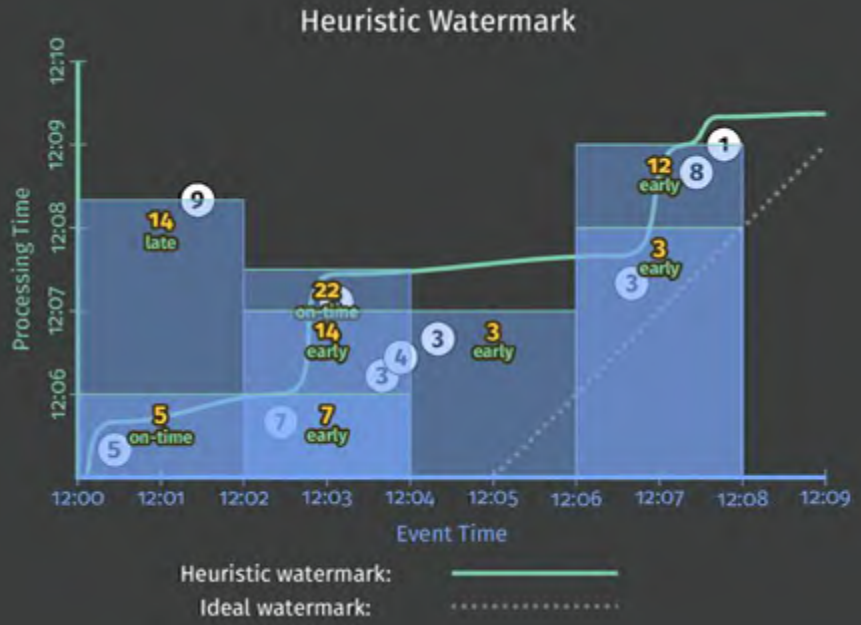


When: 提早输出(Early Firing) 和 迟到输出(Late Firing)

```
PCollection<KV<String, Integer>> scores = input
    .apply(Window
        .into(FixedWindows.of(Duration.standardMinutes(2))
        .triggering(AtWatermark()
            .withEarlyFirings(AtPeriod(Duration.standardMinutes(1))
            .withLateFirings(AtCount(1))))
    .apply(Sum.integersPerKey());
```

When: 提早输出(Early Firing) 和 迟到输出(Late Firing)





How: 如何更新结果？

- 同一个窗口的多次输出是如何累计的？
- 取决于下游消费者

Firing	Elements	Discarding	Accumulating	Acc. & Retracting
提前输出	[3]	3	3	3
按时输出	[5, 1]	6	9	(9, -3)
迟到输出	[2]	2	11	(11, -9)
Last Observed		***2***	***11***	***11***
Total Observed		***11***	***23***	***11***

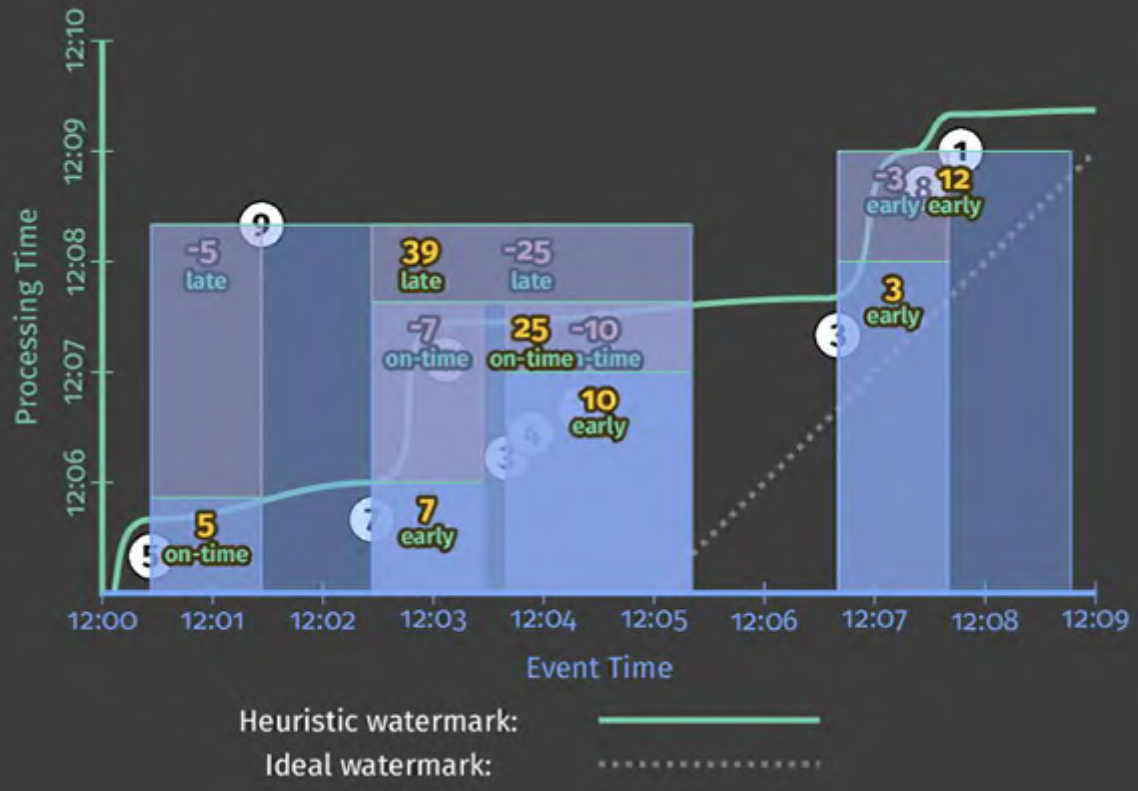
(Accumulating & Retracting not yet implemented.)

How: 加上新的，减去旧的

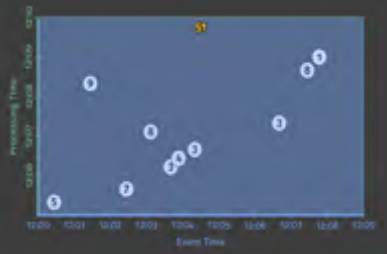
```
PCollection<KV<String, Integer>> scores = input
    .apply(Window

.into(Sessions.withGapDuration(Duration.standardMinutes(1)))
       .triggering(AtWatermark()
          .withEarlyFirings(AtPeriod(Duration.standardMinutes(1)))
          .withLateFirings(AtCount(1)))
       .accumulatingAndRetractingFiredPanes())
   .apply(Sum.integersPerKey());
```

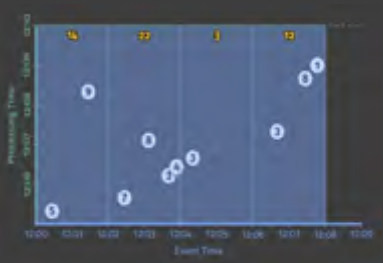
How: 加上新的，减去旧的



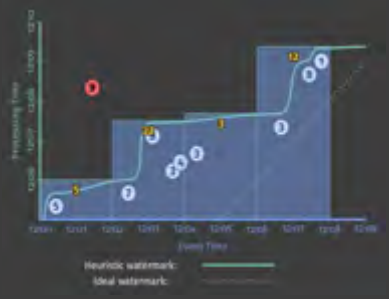
What Where When How



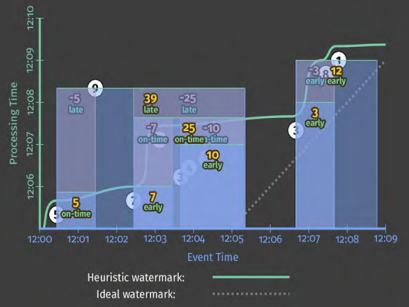
1.传统批处理



2. 支持Window的批处理



3. 流数据处理



4. 支持Retractions的流处理

Apache Beam流水线的可移植性

数据流水线开发的权衡

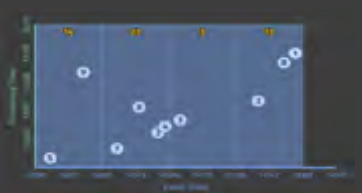
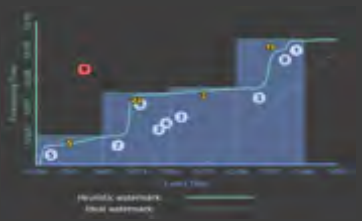
1+1=2

完整性

实时性

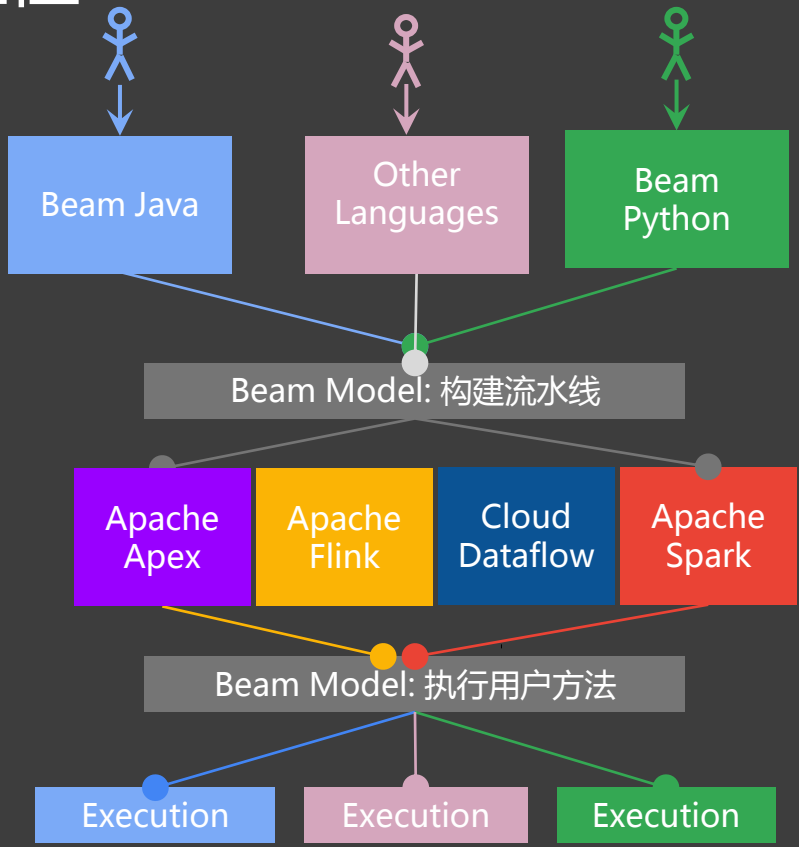
$$$

运维成本


Apache Beam流水线的可移植性

1. 编写
   - 选择最熟悉的语言*
2. 执行
   - 选择最适合的引擎
   - 统一的语义便于引擎的切换



执行引擎对Beam功能的支持表

What is being computed?

	Beam Model	Google Cloud Dataflow	Apache Flink	Apache Spark	Apache Apex
ParDo	✓	✓	✓	✓	✓
GroupByKey	✓	✓	✓	~	✓
Flatten	✓	✓	✓	✓	✓
Combine	✓	✓	✓	✓	✓
Composite Transforms	✓	~	~	~	~
Side Inputs	✓	✓	✓	✓	✓
Source API	✓	✓	✓	✓	✓
Aggregators	~	~	~	~	×
Stateful Processing	✓	~	~	×	×

Where in event time?

	Beam Model	Google Cloud Dataflow	Apache Flink	Apache Spark	Apache Apex
Global windows	✓	✓	✓	✓	✓
Fixed windows	✓	✓	✓	✓	✓
Sliding windows	✓	✓	✓	✓	✓
Session windows	✓	✓	✓	✓	✓
Custom windows	✓	✓	✓	✓	✓
Custom merging windows	✓	✓	✓	✓	✓
Timestamp control	✓	✓	✓	✓	✓

When in processing time?

	Beam Model	Google Cloud Dataflow	Apache Flink	Apache Spark	Apache Apex
Configurable triggering	✓	✓	✓	×	✓
Event-time triggers	✓	✓	✓	×	✓
Processing-time triggers	✓	✓	✓	✓	✓
Count triggers	✓	✓	✓	×	✓
[Meta]data driven triggers	× (BEAM-101)	×	×	×	×
Composite triggers	✓	✓	✓	×	✓
Allowed lateness	✓	✓	✓	×	✓
Timers	✓	~	~	×	×

How do refinements relate?

	Beam Model	Google Cloud Dataflow	Apache Flink	Apache Spark	Apache Apex
Discarding	✓	✓	✓	✓	✓
Accumulating	✓	✓	✓	×	✓
Accumulating & Retracting	× (BEAM-91)	×	×	×	×

https://beam.apache.org/learn/runners/capability-matrix/

Apache Beam的高效性

Apache Beam与阿里巴巴

Growing the Community

Apache Beam交流群

- 如果你有大数据APIs, 写一个Beam **SDK** 或者 **DSL**.
- 如果你有一个存储系统、消息系统，写一个Beam **IO connector**!
- 如果你有一个分布式引擎，写一个Beam **runner**!

谢谢!

Learn More!

Apache Beam

https://beam.apache.org

Resources

https://beam.apache.org/documentation/resources/

Join the Beam mailing lists!

user-subscribe@beam.apache.org
dev-subscribe@beam.apache.org