FLINK FORWARD

Berlin, September 3-5, 2018

# Efficient Window Aggregation with Stream Slicing


**Jonas Traub**
Research Associate (TU Berlin)



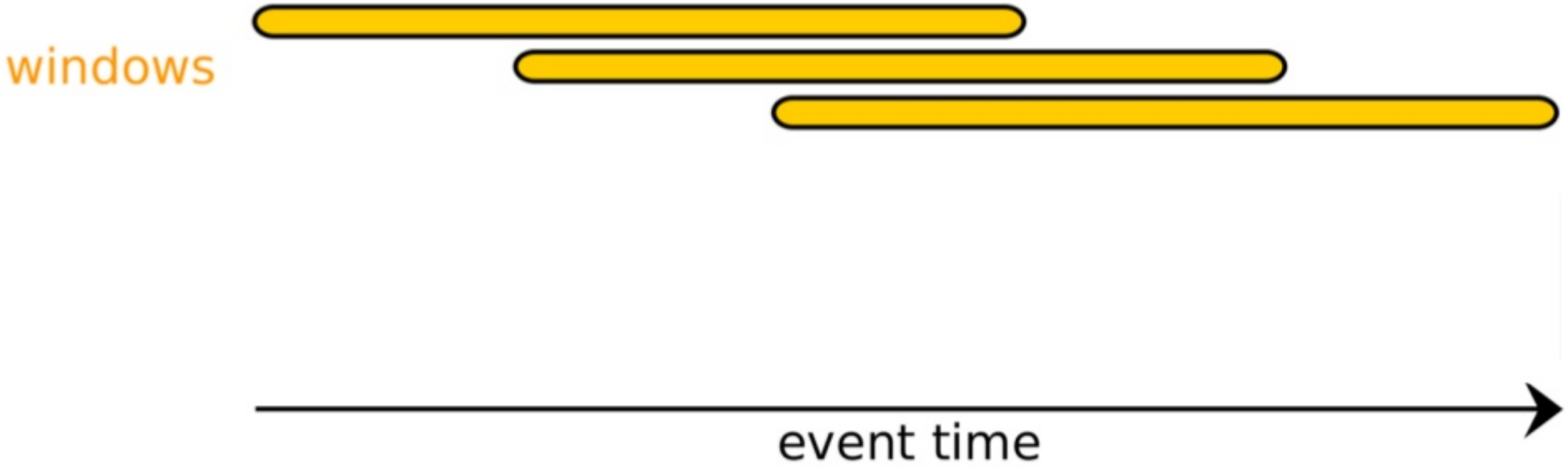
**Philipp M. Grulich**
Research Assistant (DFKI)

German Research Center for Artificial Intelligence

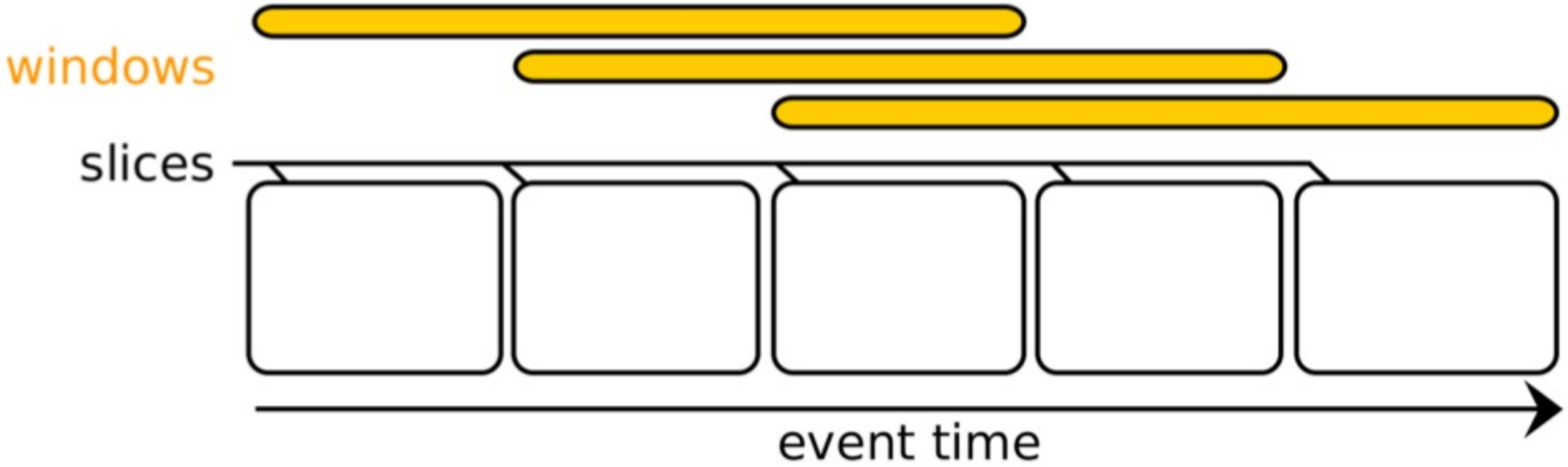
FLINK FORWARD

# Stream Slicing Example

# Stream Slicing Example

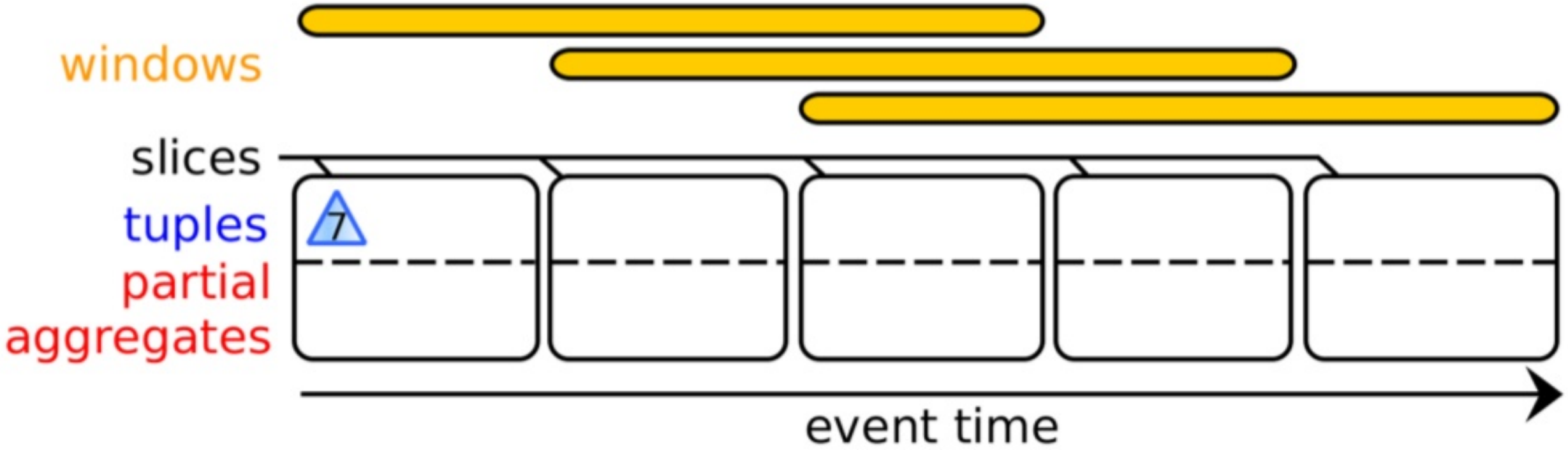


# Stream Slicing Example



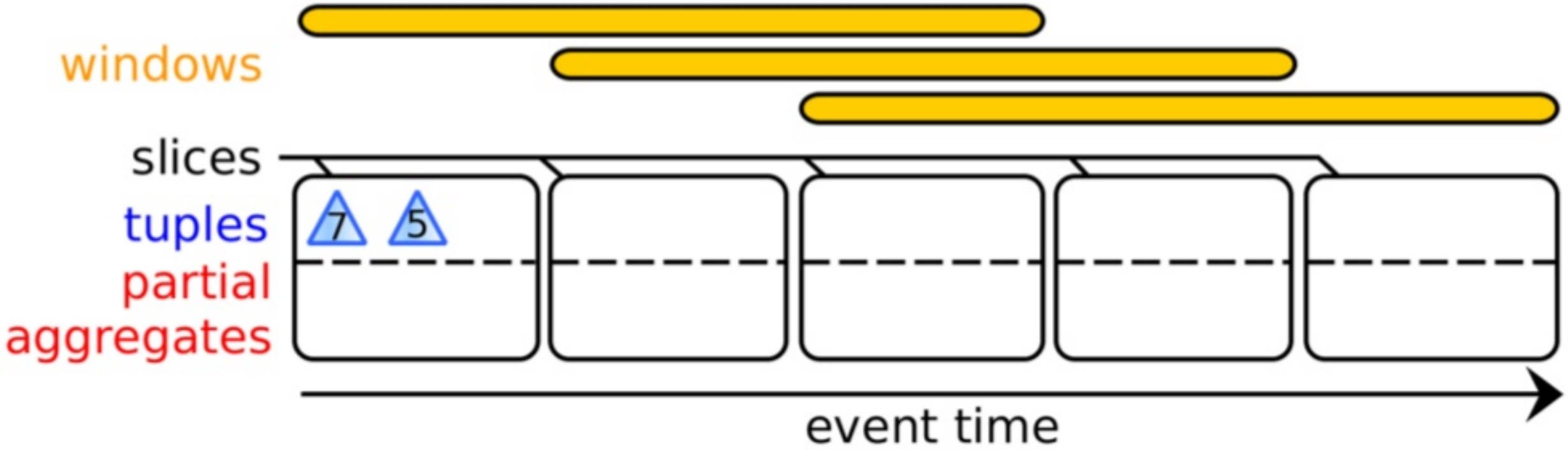
FLINK FORWARD

# Stream Slicing Example



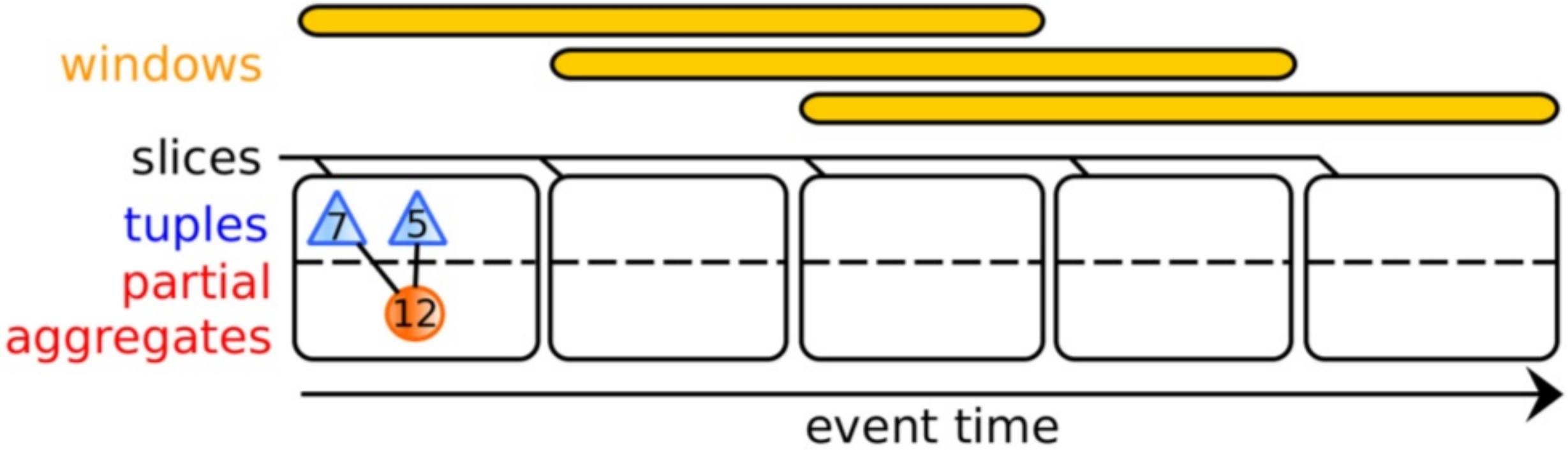
FLINK FORWARD

# Stream Slicing Example



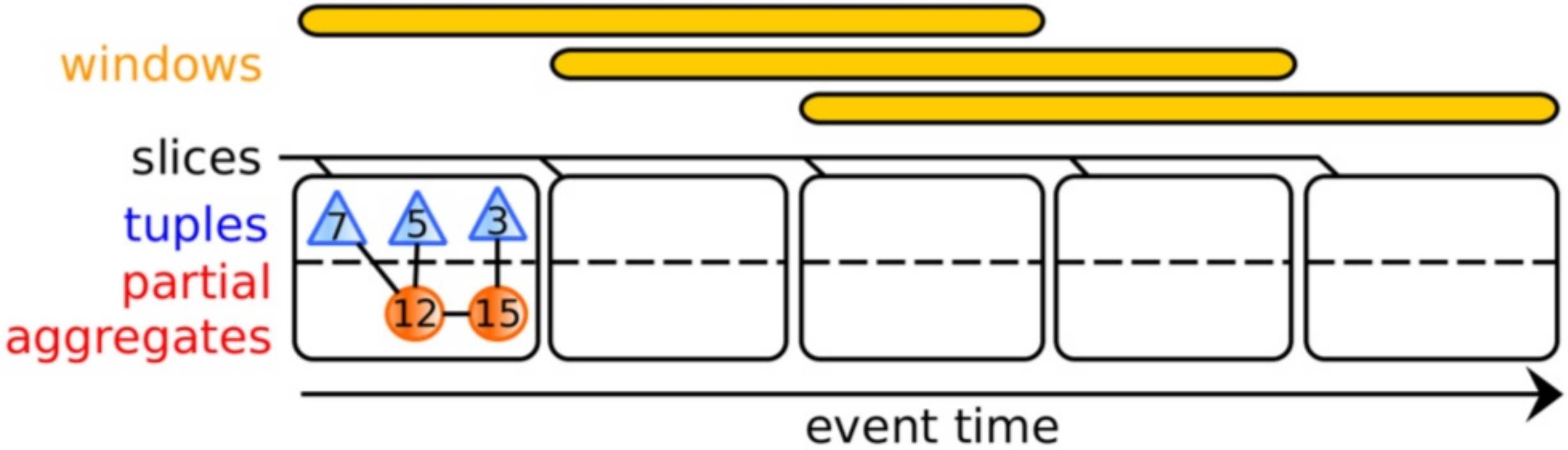
FLINK FORWARD

# Stream Slicing Example



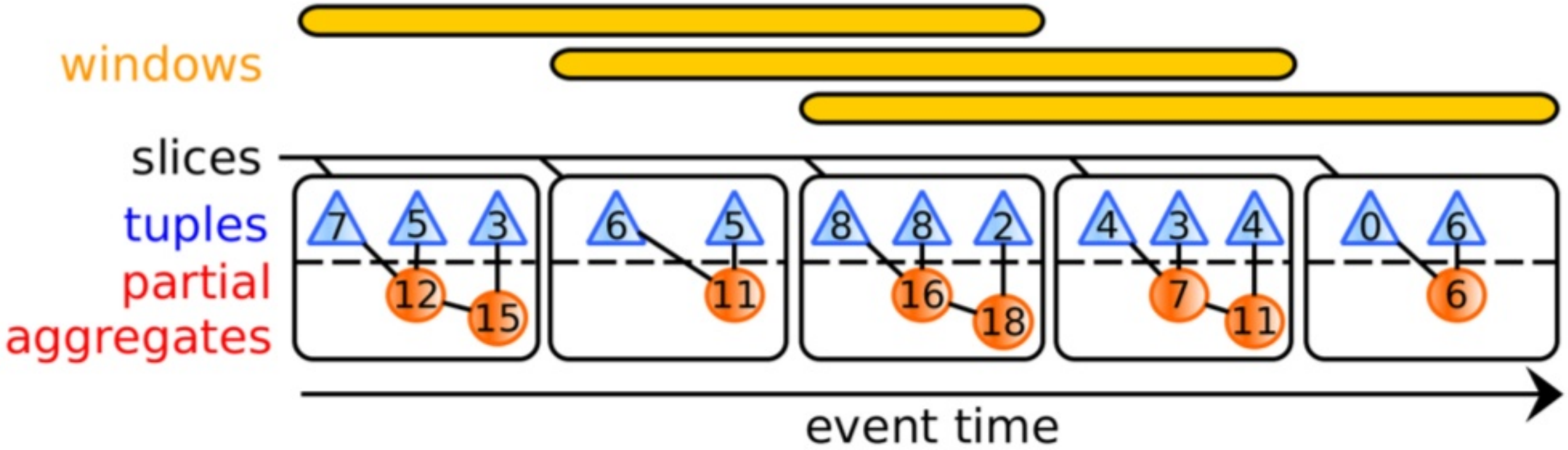
FLINK FORWARD

# Stream Slicing Example

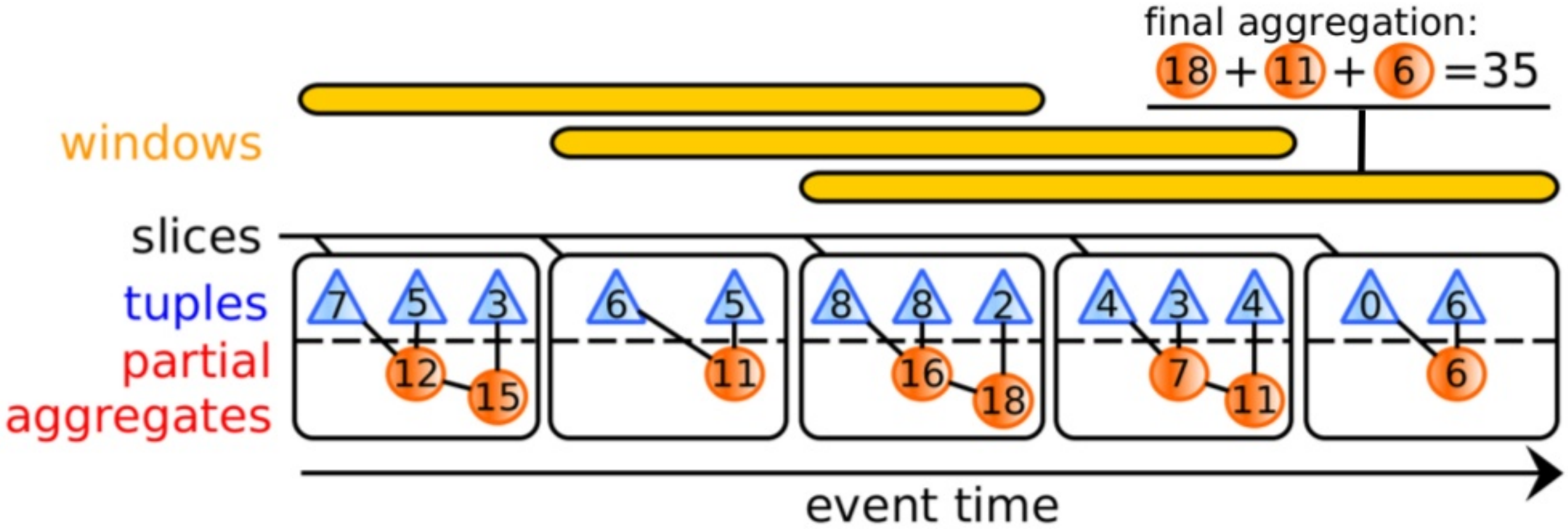


FLINK FORWARD

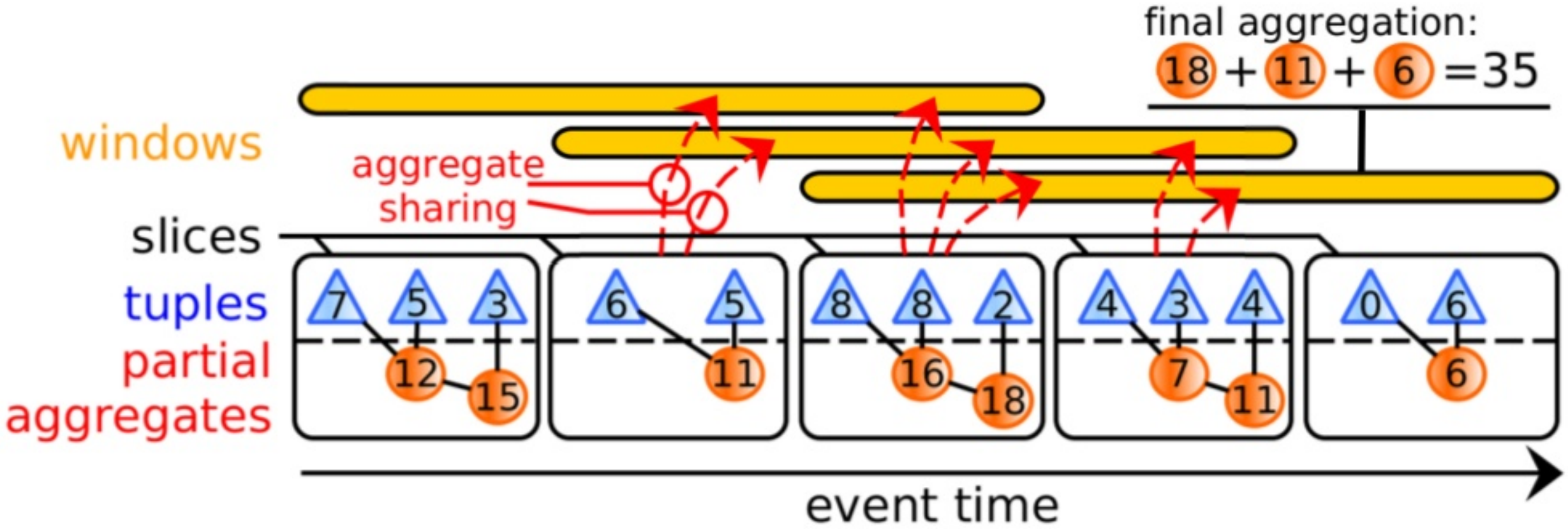
# Stream Slicing Example



# Stream Slicing Example



# Stream Slicing Example



FLINK FORWARD

# Stream Slicing Research

# Stream Slicing Research

CIKM 2016

## Cutty: Aggregate Sharing for User-Defined Windows

Paris Carbone† Jonas Traub‡ Asterios Katsifodimos‡
Seif Haridi† Volker Markl‡

# Stream Slicing Research

CIKM 2016

**Cutty: Aggregate Sharing for User-Defined Windows**

Paris Carbone[†] Jonas Traub[‡] Asterios Katsifodimos[‡]
Seif Haridi[†] Volker Markl[‡]

ICDE 2018

**Scotty: Efficient Window Aggregation for out-of-order Stream Processing**

Jonas Traub[1], Philipp M. Grulich[2], Alejandro Rodríguez Cuéllar[1], Sebastian Breß[1,2]
Asterios Katsifodimos[3], Tilmann Rabl[1,2], Volker Markl[1,2]

# Flink Windowing Bottlenecks

Example Query:

```
.window(SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10)))
.sum()
```

# Flink Windowing Bottlenecks

Example Query:

```
.window(SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10)))
.sum()
```

Processing with Buckets:

# Flink Windowing Bottlenecks

Example Query:

```
.window(SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10)))
.sum()
```

Processing with Buckets:

# Flink Windowing Bottlenecks

Example Query:

```
.window(SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10)))
.sum()
```

Processing with Buckets:

# Flink Windowing Bottlenecks

Example Query:

```
.window(SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10)))
.sum()
```

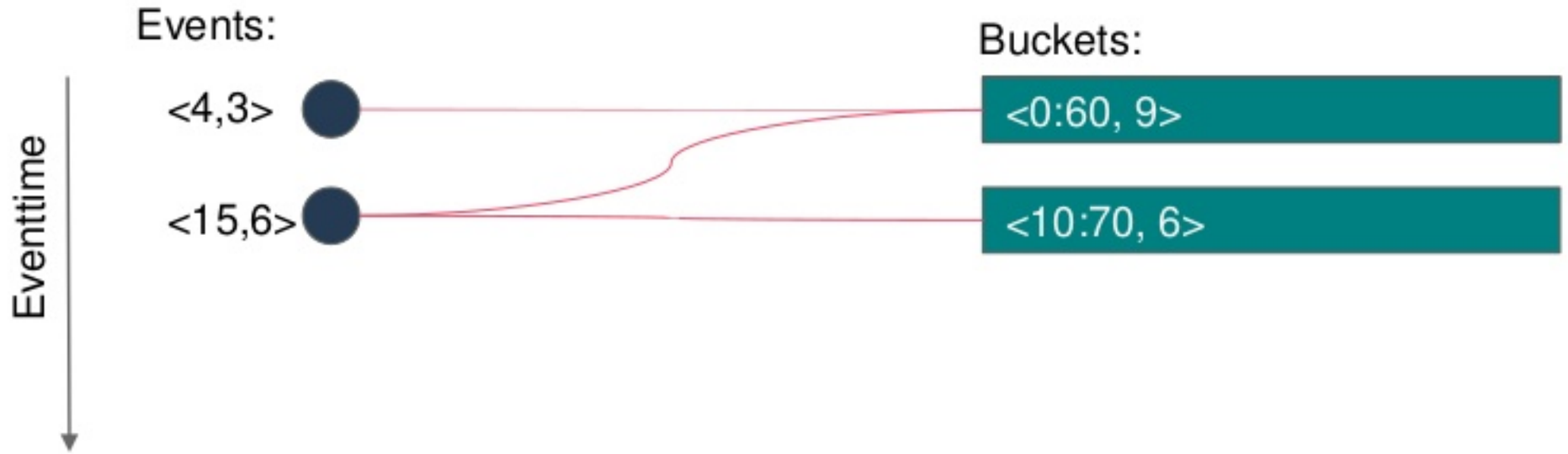
Processing with Buckets:

# Flink Windowing Bottlenecks

Example Query:

```
.window(SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10)))
.sum()
```

Processing with Buckets:

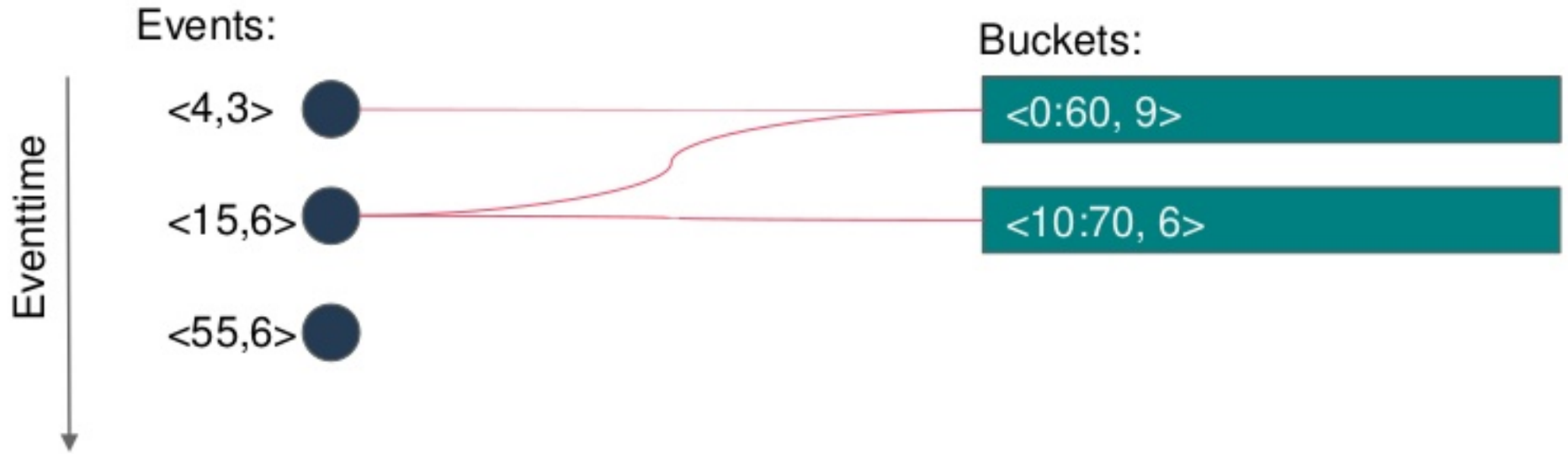


# Flink Windowing Bottlenecks

Example Query:

```
.window(SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10)))
.sum()
```

Processing with Buckets:

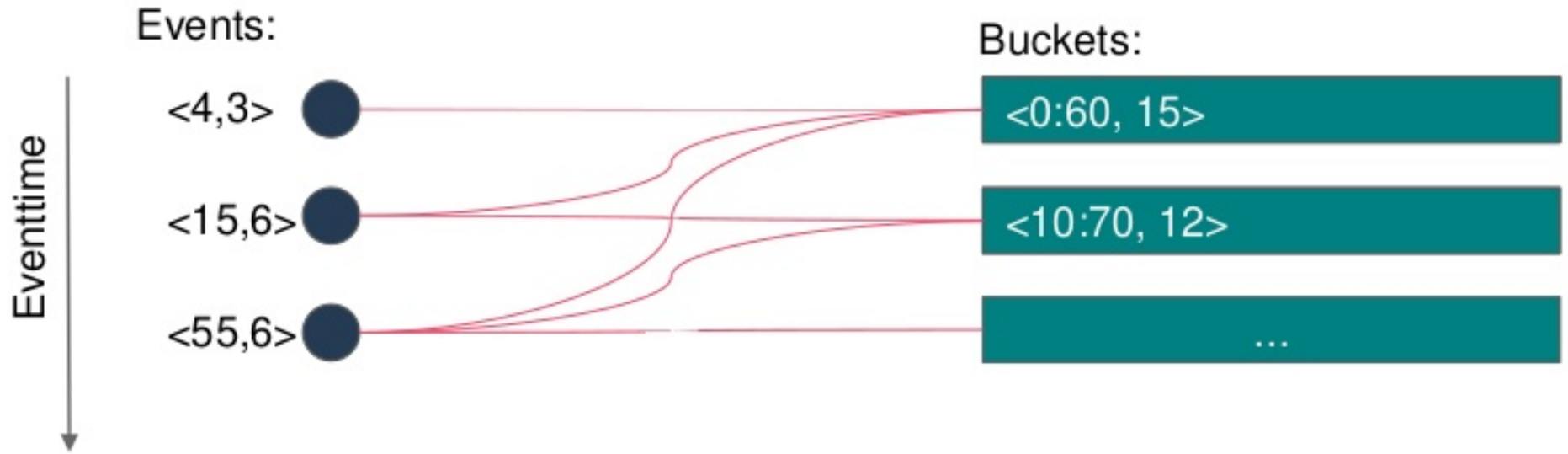


# Flink Windowing Bottlenecks

Example Query:

```
.window(SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10)))
.sum()
```

Processing with Buckets:

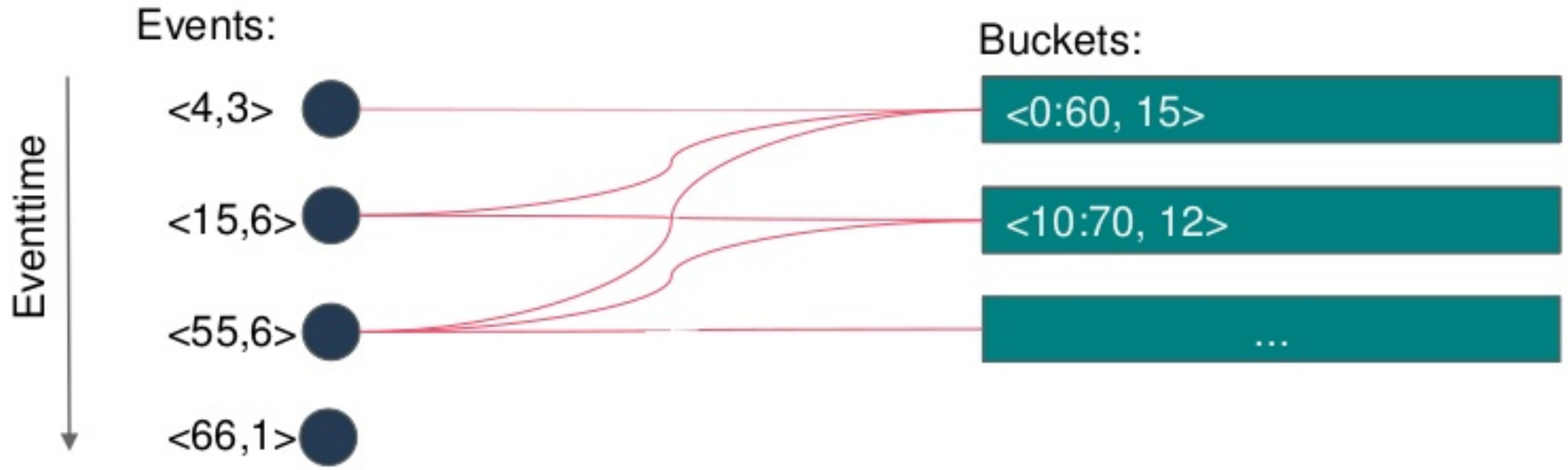


# Flink Windowing Bottlenecks

Example Query:

```
.window(SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10)))
.sum()
```

Processing with Buckets:

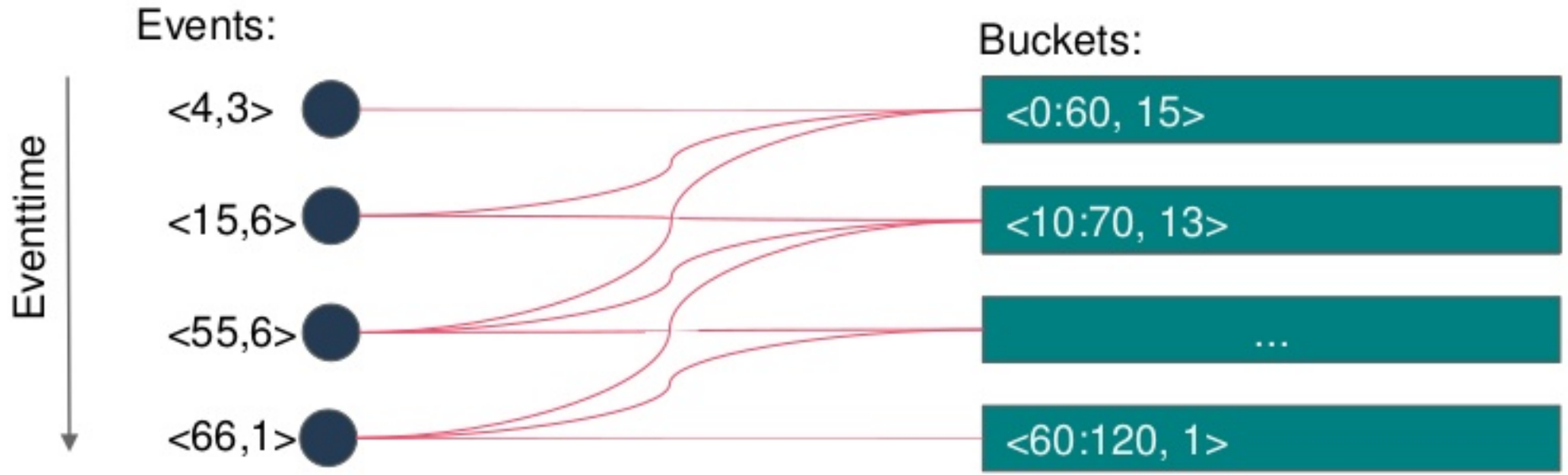


# Flink Windowing Bottlenecks

Example Query:

```
.window(SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10)))
.sum()
```

Processing with Buckets:



FLINK FORWARD

# Flink Windowing Bottlenecks

Number of Buckets = Window Length / Slide Length

# Flink Windowing Bottlenecks

Number of Buckets = Window Length / Slide Length

```
SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10))
```

--> 6 Buckets

# Flink Windowing Bottlenecks

Number of Buckets = Window Length / Slide Length

```
SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10))
```

--> 6 Buckets

```
SlidingEventTimeWindows.of(Time.day(1), Time.seconds(10))
```

--> 8640 Buckets

# Flink Windowing Bottlenecks

Number of Buckets = Window Length / Slide Length

```
SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10))
```

--> 6 Buckets

```
SlidingEventTimeWindows.of(Time.day(1), Time.seconds(10))
```

--> 8640 Buckets

Overlapping windows cause:

# Flink Windowing Bottlenecks

Number of Buckets = Window Length / Slide Length

```
SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10))
```

--> 6 Buckets

```
SlidingEventTimeWindows.of(Time.day(1), Time.seconds(10))
```

--> 8640 Buckets

Overlapping windows cause:

- Every event is assigned to many windows.

# Flink Windowing Bottlenecks

Number of Buckets = Window Length / Slide Length

```
SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10))
```

--> 6 Buckets

```
SlidingEventTimeWindows.of(Time.day(1), Time.seconds(10))
```

--> 8640 Buckets

Overlapping windows cause:

- Every event is assigned to many windows.
- Repeated aggregations --> aggregation function is called on every window

# Flink Windowing Bottlenecks

Number of Buckets = Window Length / Slide Length

```
SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10))
```
--> 6 Buckets

```
SlidingEventTimeWindows.of(Time.day(1), Time.seconds(10))
```
--> 8640 Buckets

Overlapping windows cause:

- Every event is assigned to many windows.
- Repeated aggregations --> aggregation function is called on every window
- High memory consumption --> especially for windows without incremental aggregation

# Flink Windowing Bottlenecks

Number of Buckets = Window Length / Slide Length

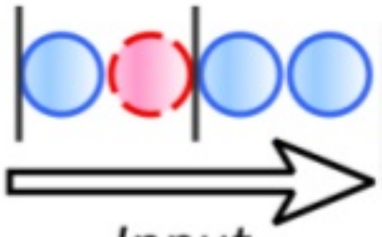
```
SlidingEventTimeWindows.of(Time.minutes(1), Time.seconds(10))
```
--> 6 Buckets

```
SlidingEventTimeWindows.of(Time.day(1), Time.seconds(10))
```
--> 8640 Buckets

Overlapping windows cause:

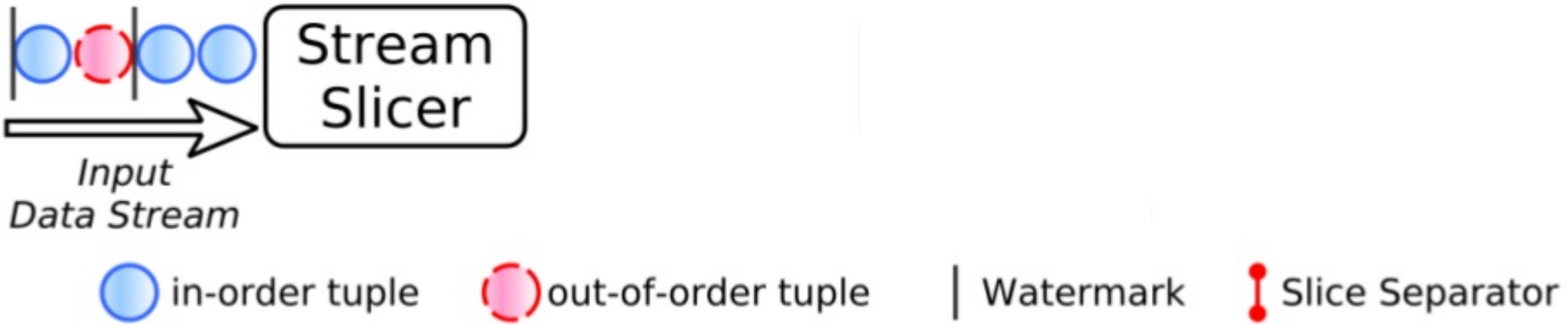
- Every event is assigned to many windows.
- Repeated aggregations --> aggregation function is called on every window
- High memory consumption --> especially for windows without incremental aggregation
- Check for merging windows

# Architecture Overview

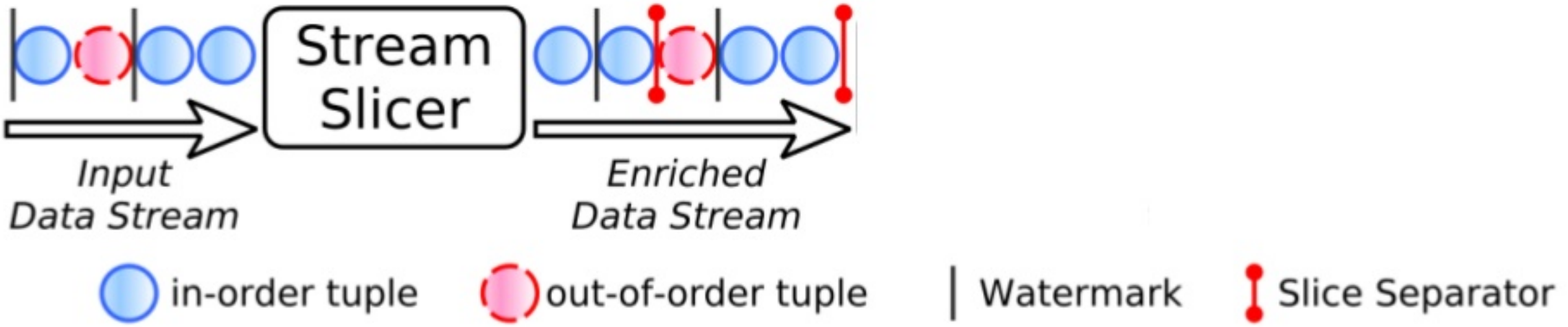



in-order tuple  out-of-order tuple  Watermark  Slice Separator

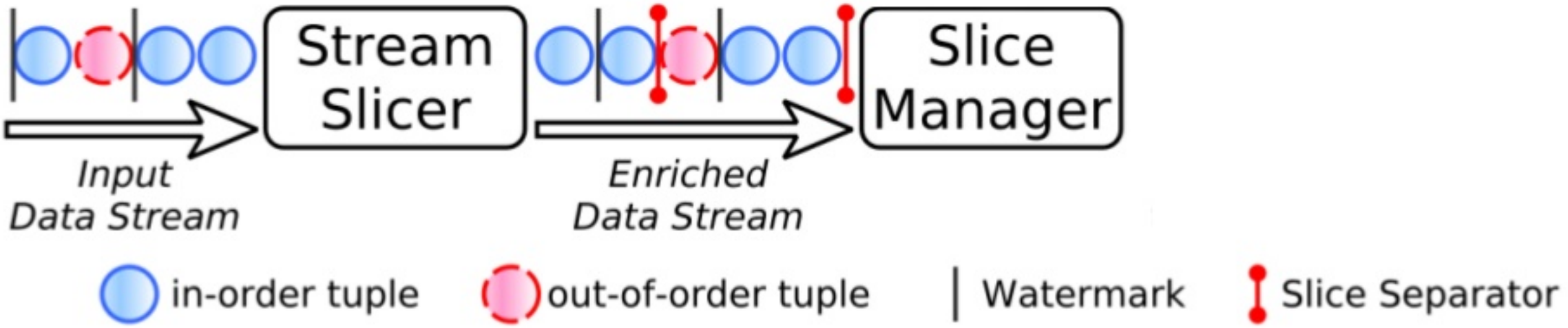
# Architecture Overview



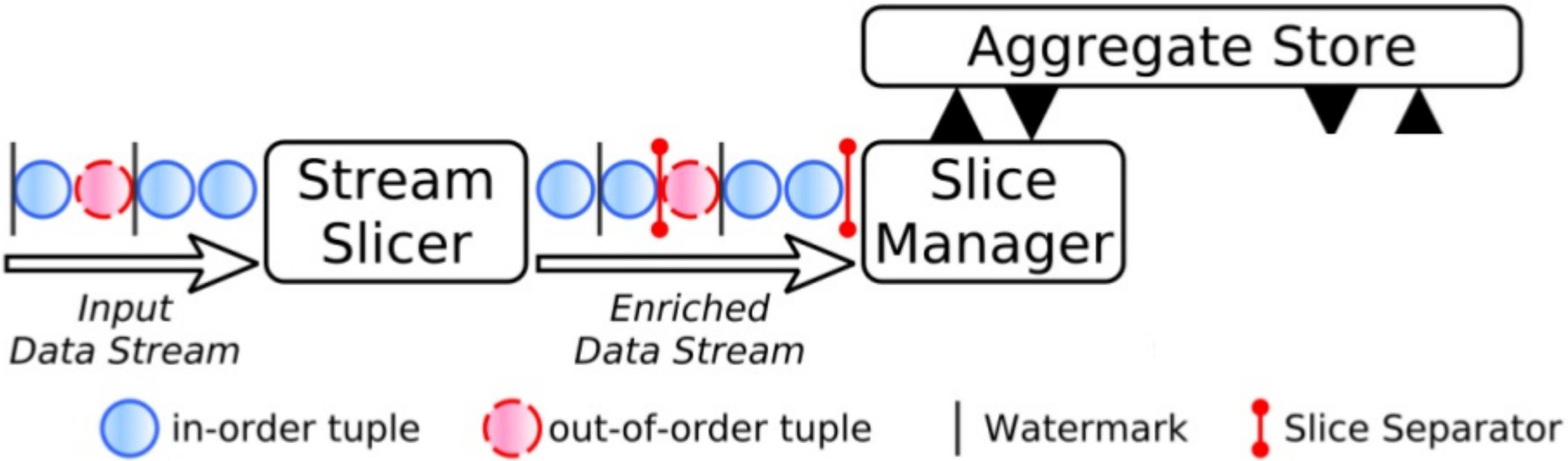
# Architecture Overview



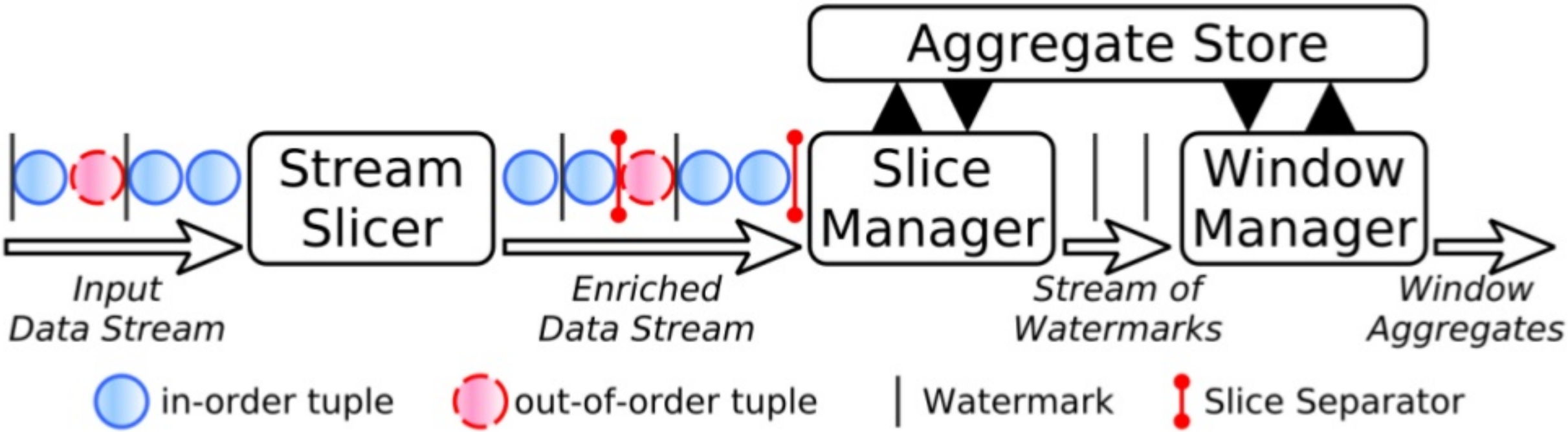
# Architecture Overview



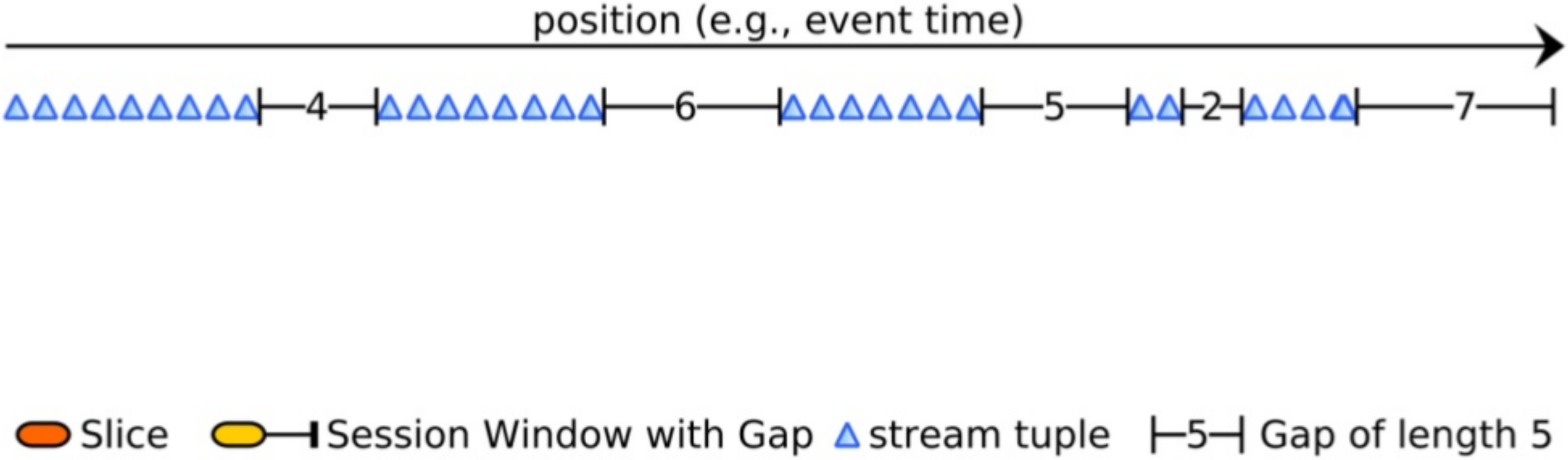
# Architecture Overview



# Architecture Overview

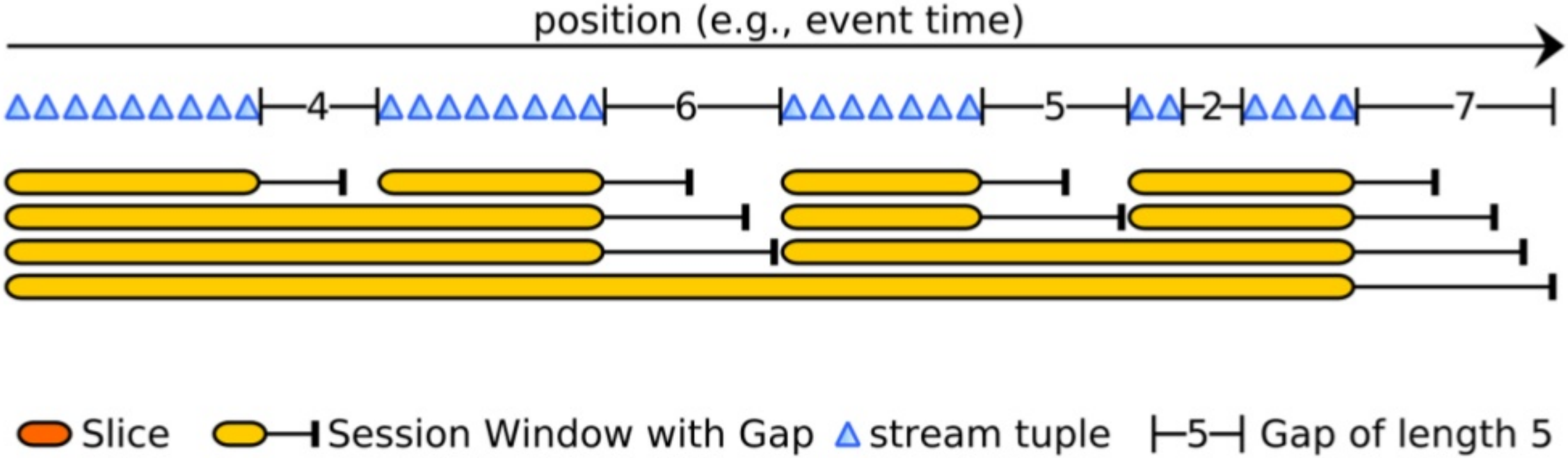


# Session Window Aggregate Sharing



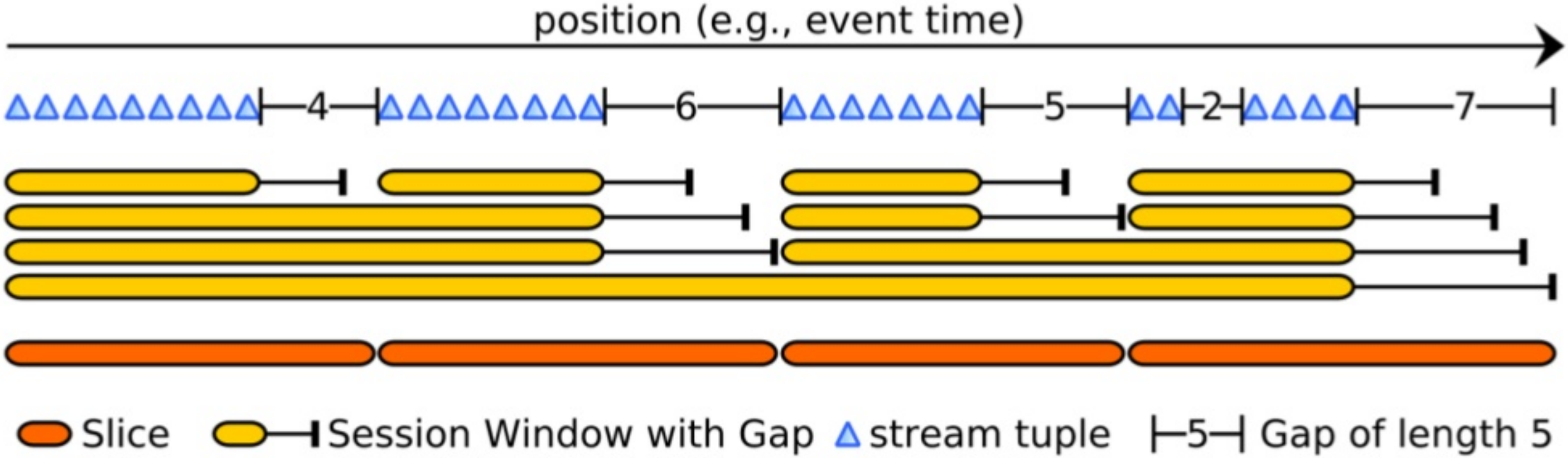
FLINK FORWARD

# Session Window Aggregate Sharing



FLINK FORWARD

# Session Window Aggregate Sharing



# Out-of-Order Processing and Sessions


1) Insert:

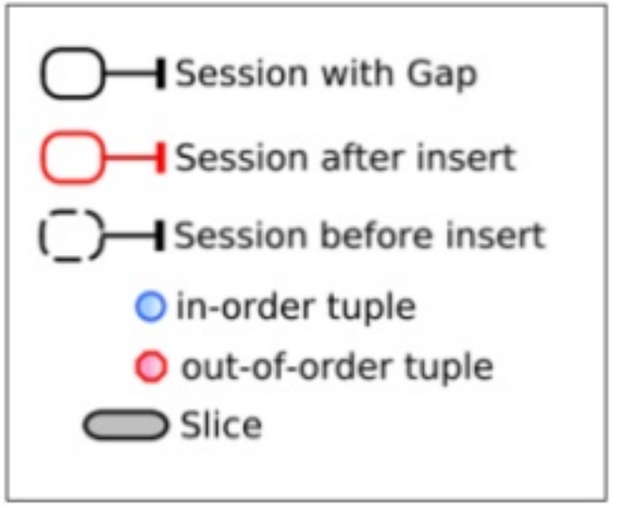
1.1) Simple Insert:

1.2) Right Insert:

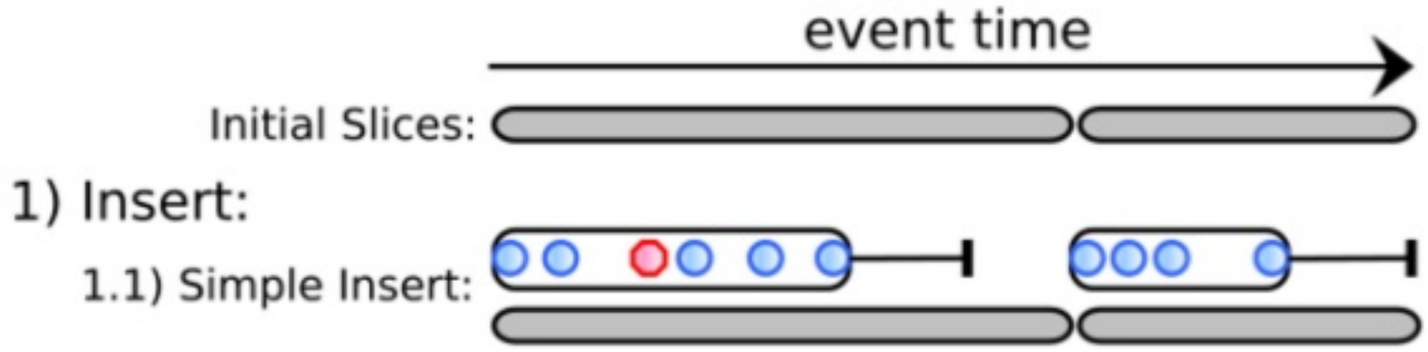
1.3) Left Insert:

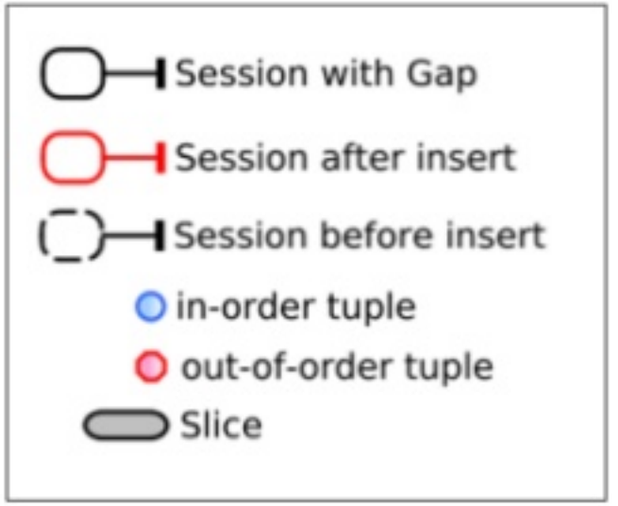
2) Fuse Sessions:

3) New Session:

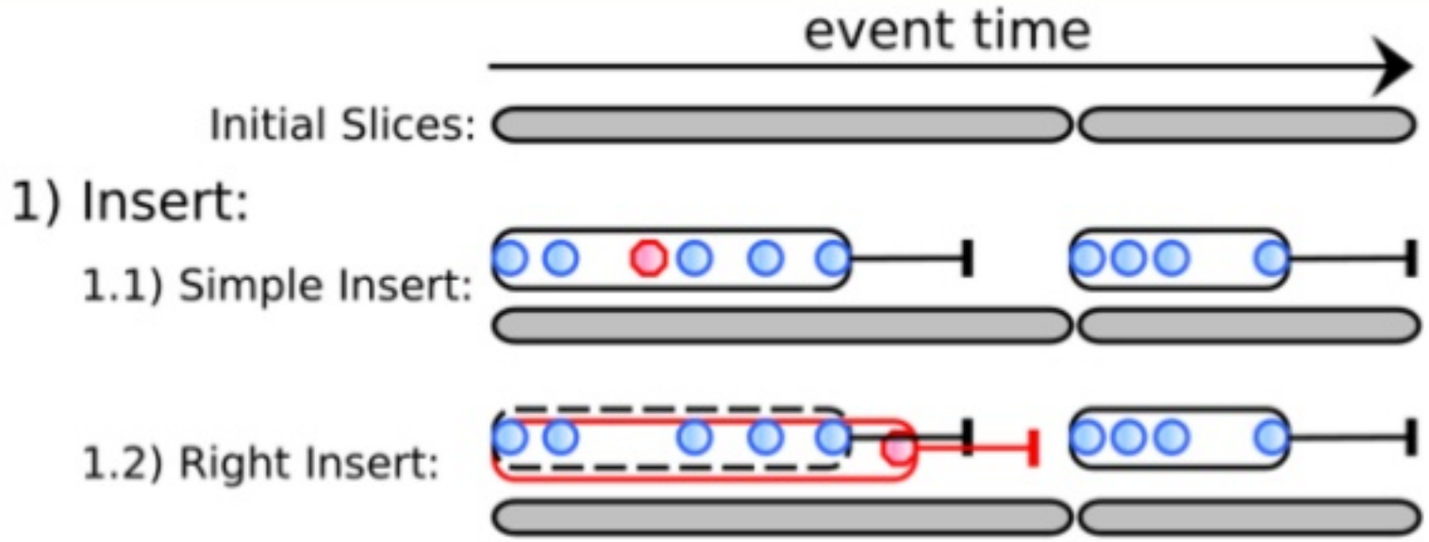


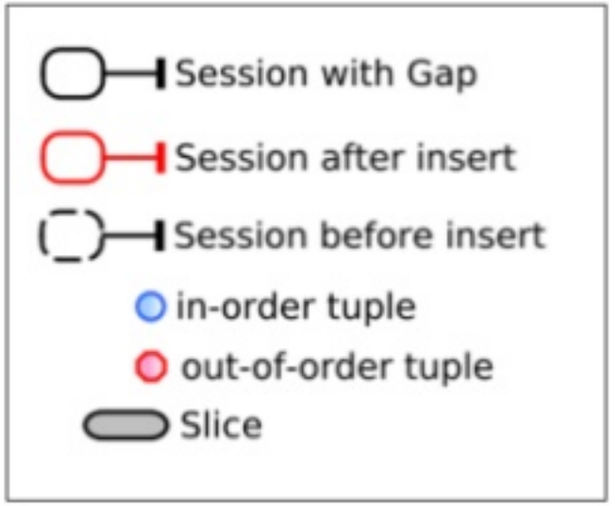
# Out-of-Order Processing and Sessions



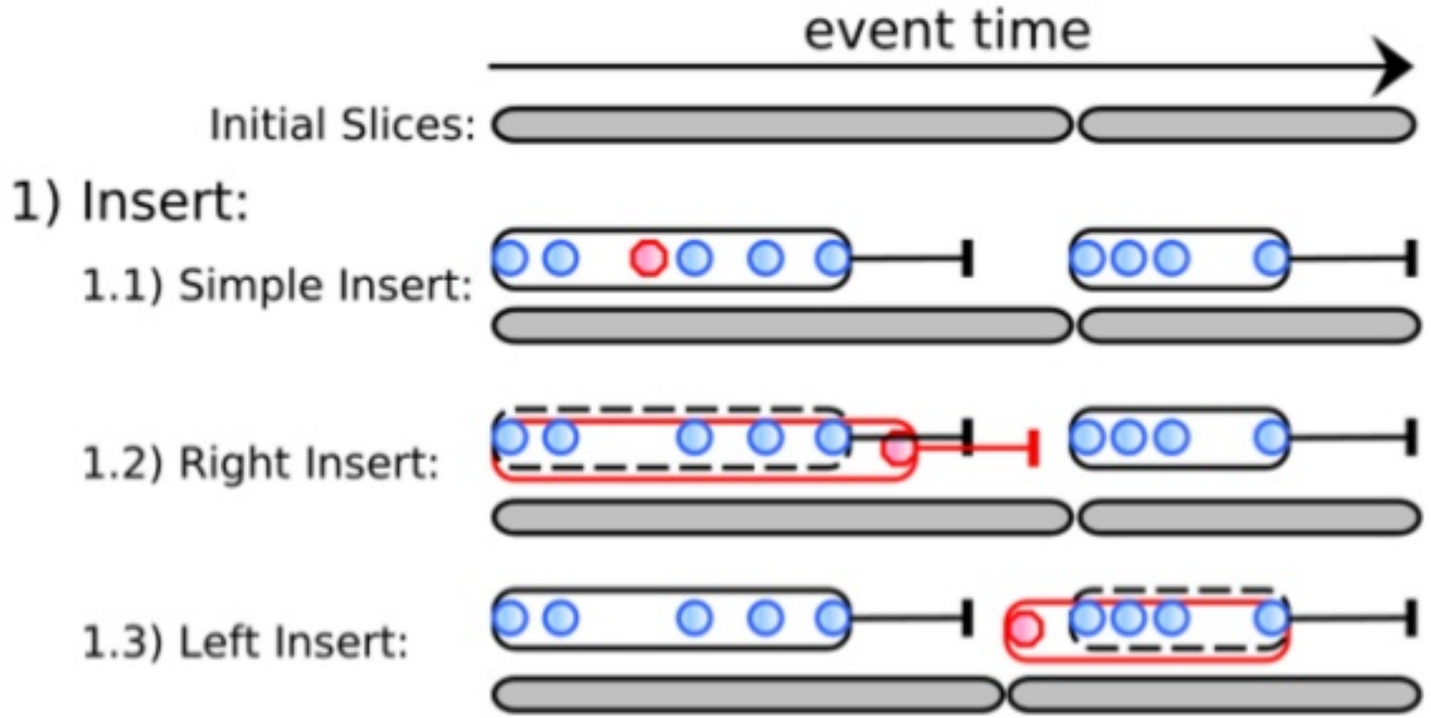



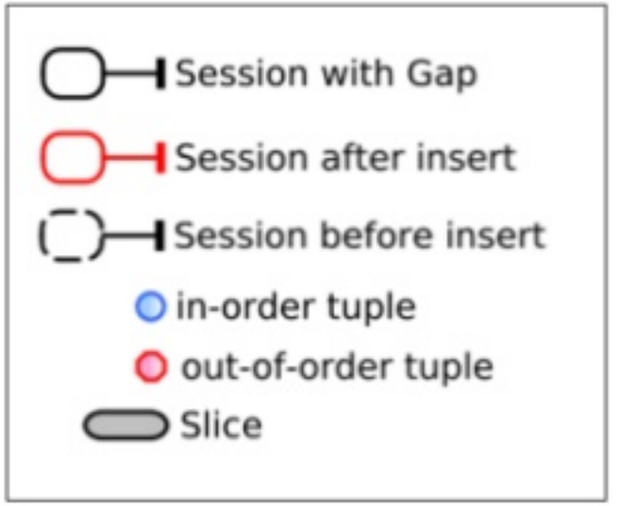
# Out-of-Order Processing and Sessions



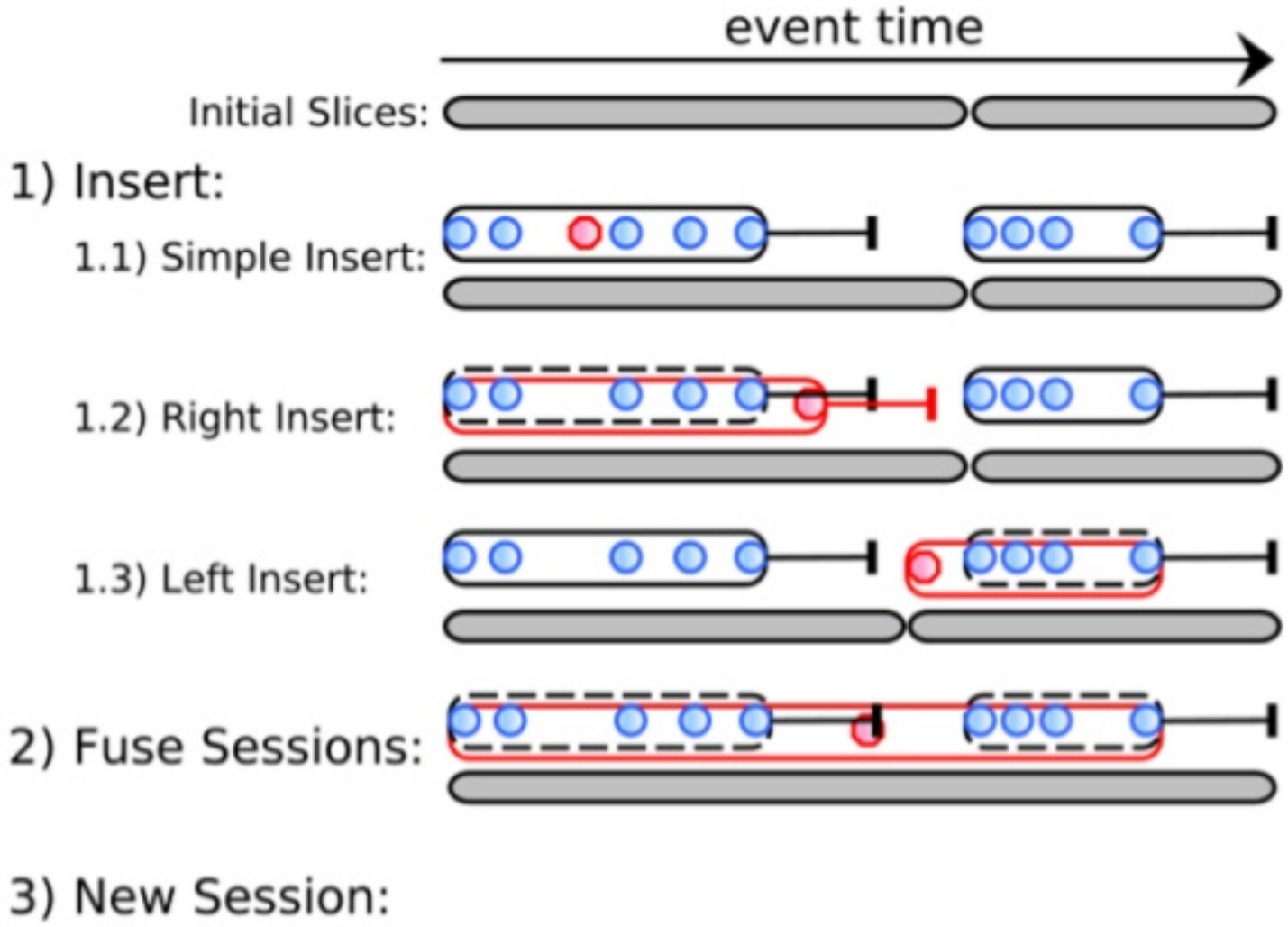



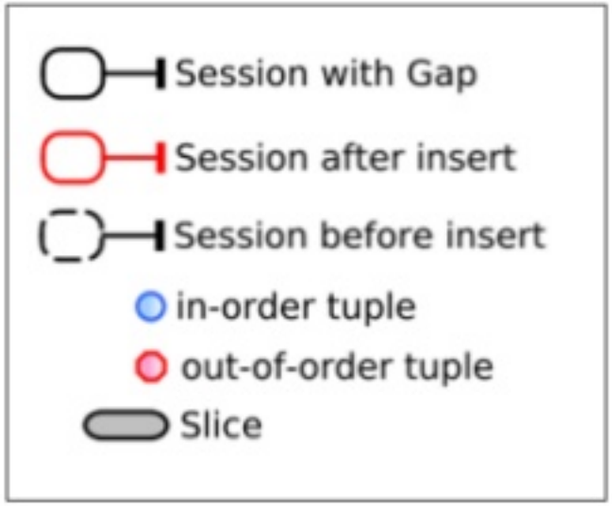
# Out-of-Order Processing and Sessions



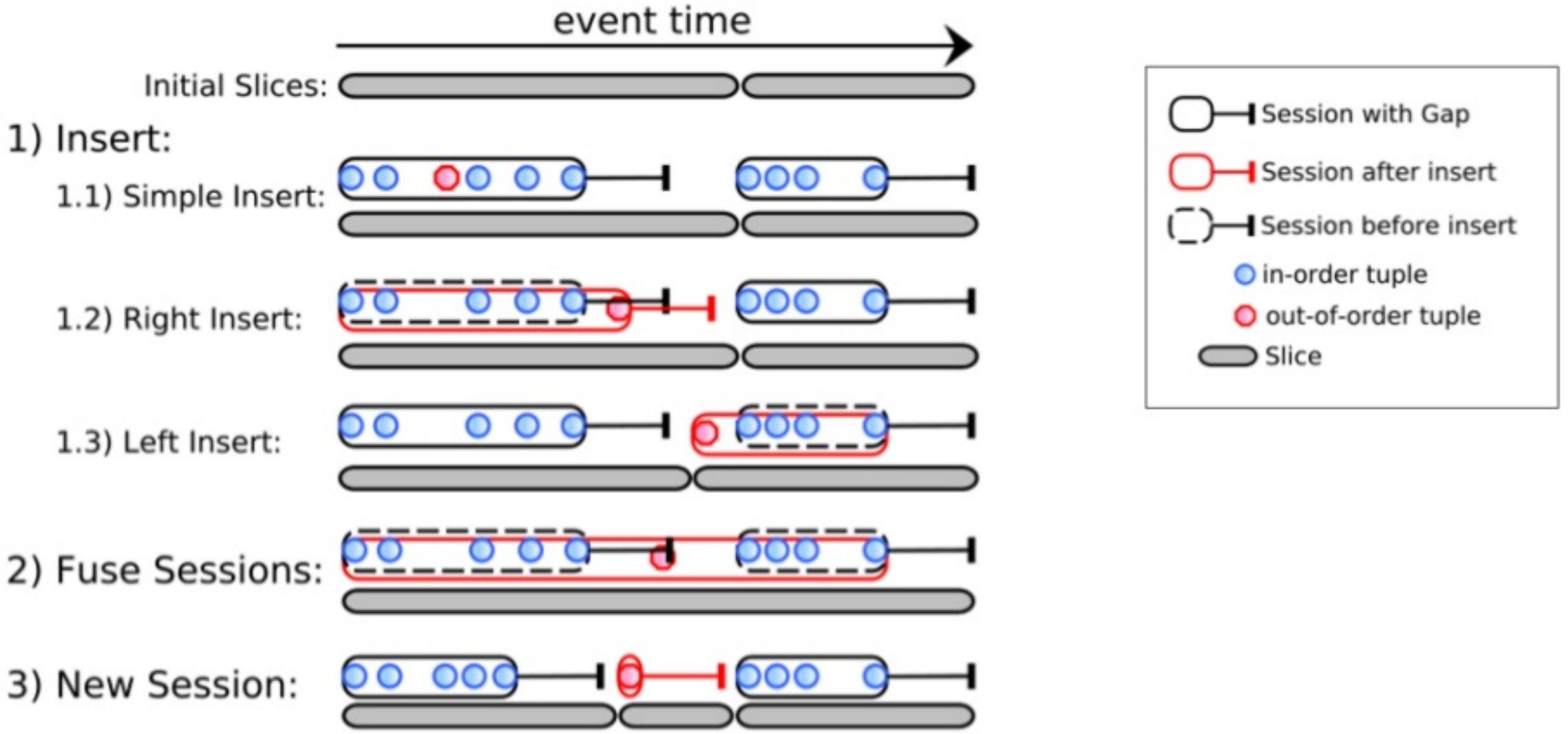



# Out-of-Order Processing and Sessions



# Out-of-Order Processing and Sessions

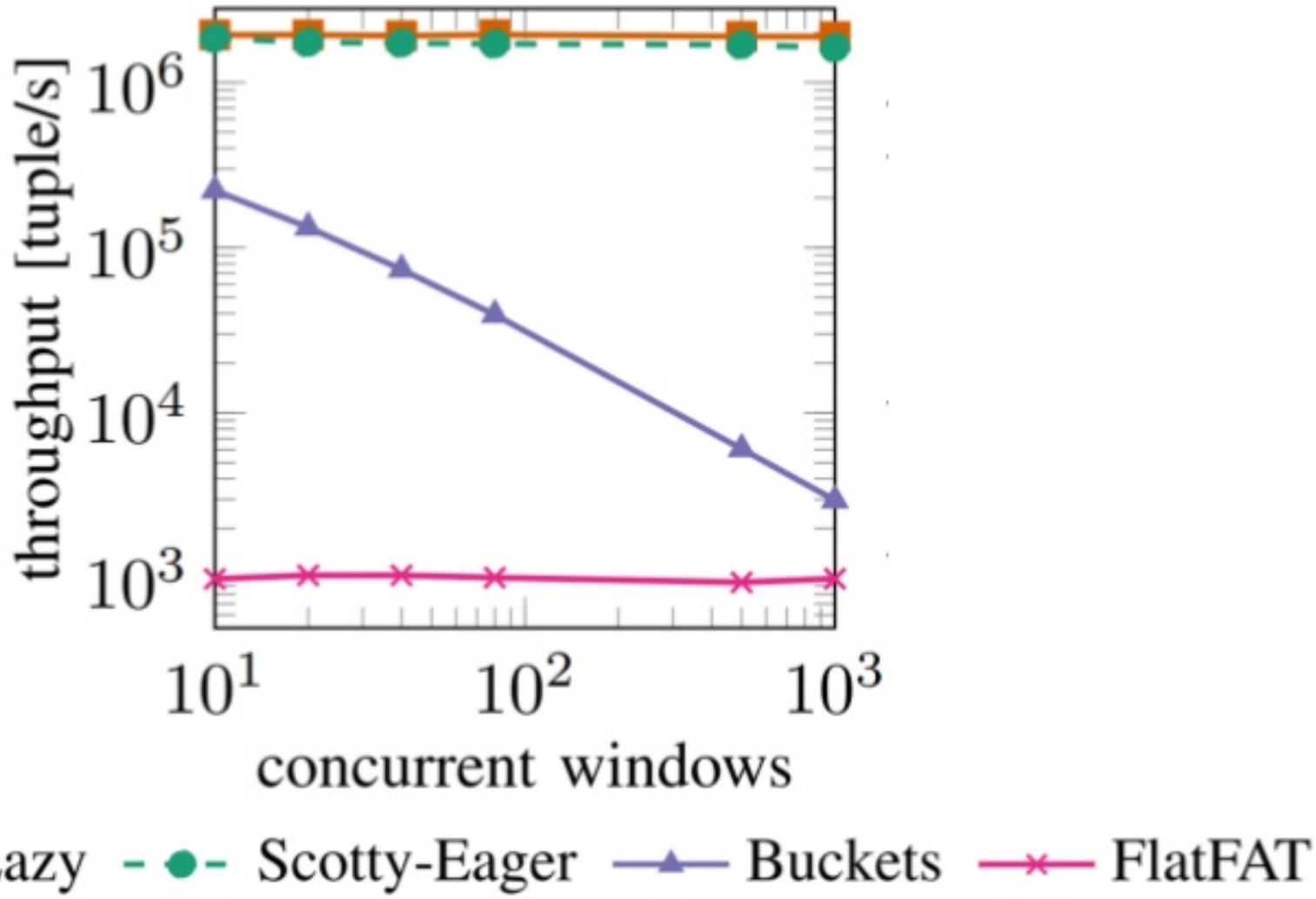


# Multi-Window Processing

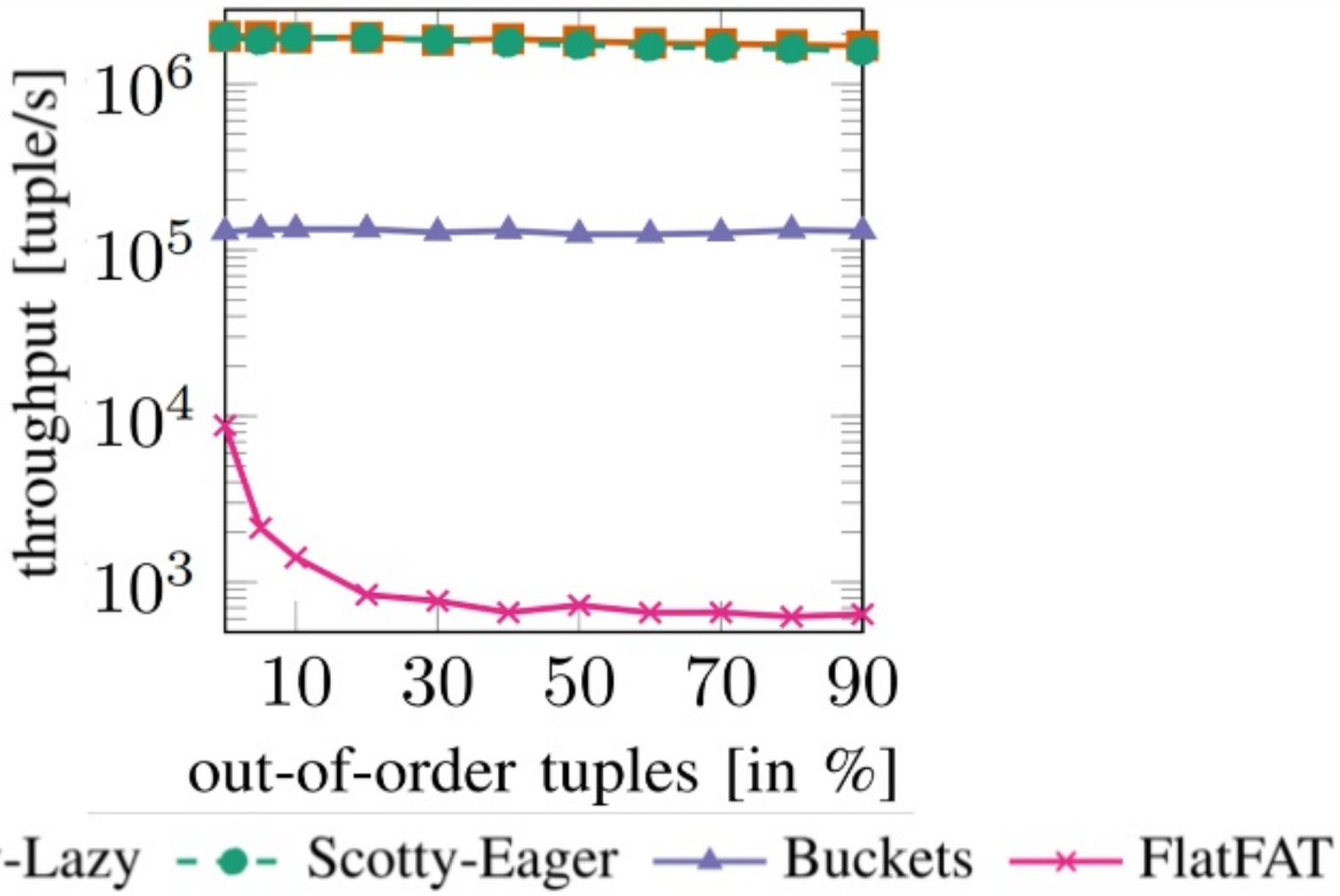
## Multi-Window Processing (Example: Fitness Tracker)

```
[...].window(
      // Daily report:
      TumblingEventTimeWindows.of(Time.days(1)),
      // Monitoring dashboard (last hour):
      SlidingEventTimeWindows.of(Time.hours(1), Time.seconds(1)),
      // Activity periods:
      EventTimeSessionWindows.of(Time.minutes(1)))
.sum()
```

# Stream Slicing Performance



# Stream Slicing Performance



FLINK FORWARD

# Runtime-Dynamic Windows

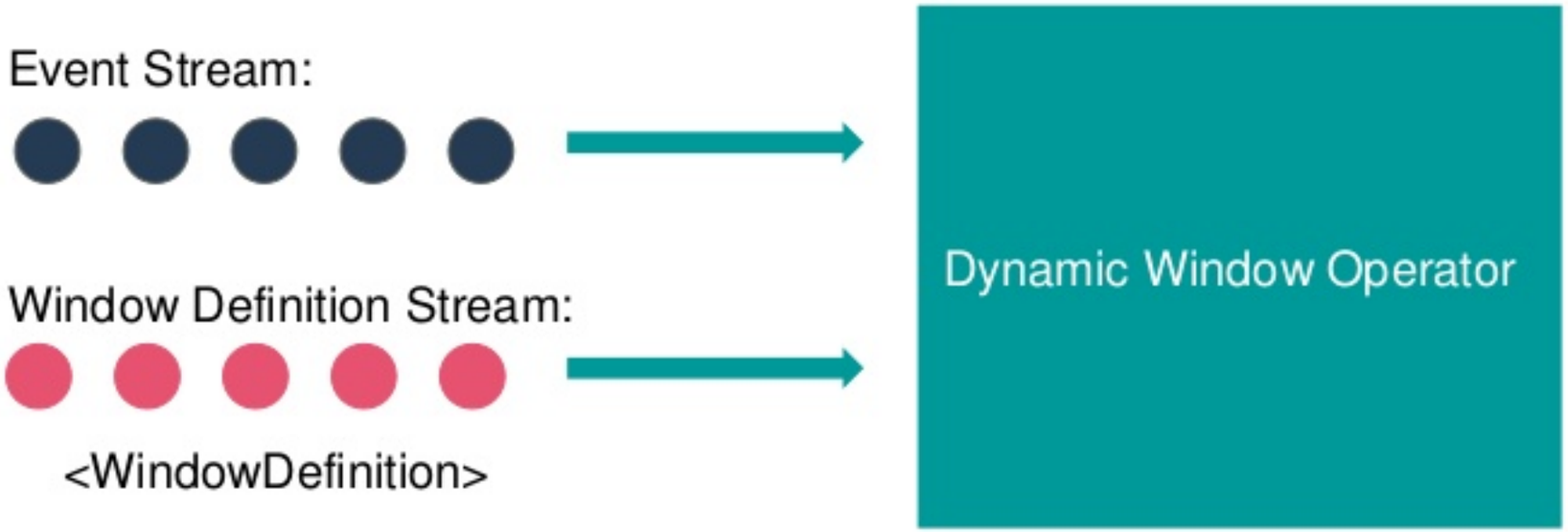
# Runtime-Dynamic Windows

Event Stream:

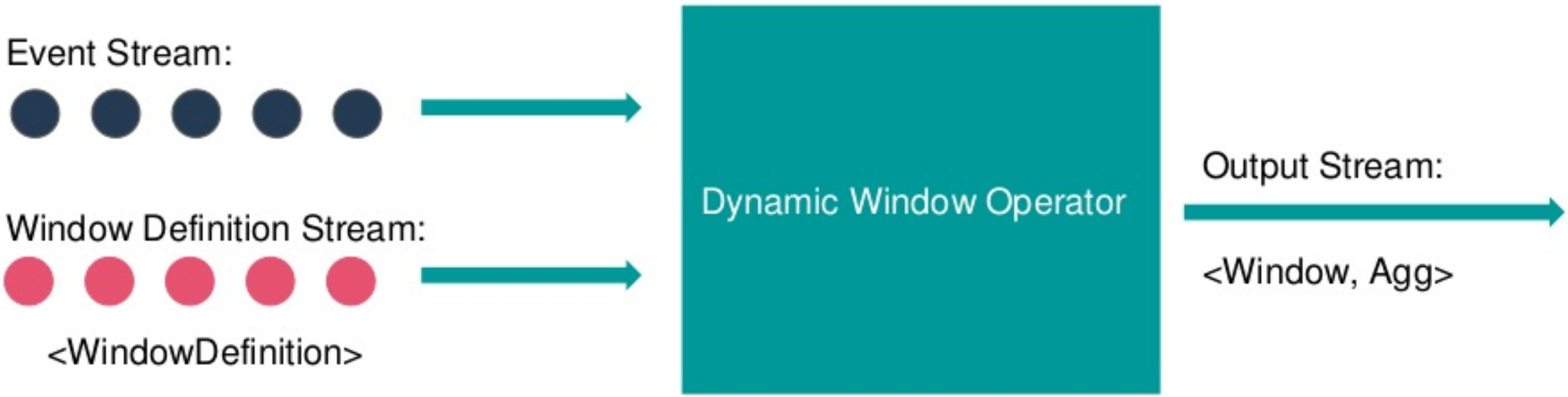
Window Definition Stream:

<WindowDefinition>

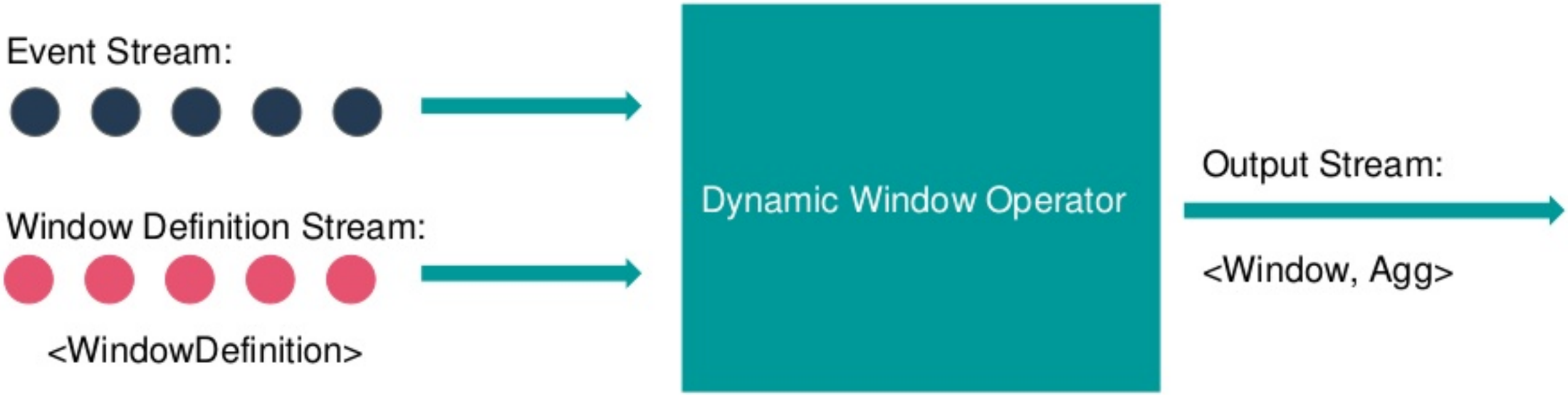
# Runtime-Dynamic Windows



# Runtime-Dynamic Windows



# Runtime-Dynamic Windows




```
.dynamicWindow(windowDefinitionStream)
.sum()
```

# From Research to Production

# From Research to Production


- Implement complete fault-tolerance and state-management

# From Research to Production


- Implement complete fault-tolerance and state-management
- State migration

# From Research to Production


- Implement complete fault-tolerance and state-management
- State migration
  - Hard limitation: Aggregated buckets in state snapshots cannot be migrated

# From Research to Production


- Implement complete fault-tolerance and state-management
- State migration
  - Hard limitation: Aggregated buckets in state snapshots cannot be migrated
- Sophisticated testing

# From Research to Production

**Research**

**Production**

- Implement complete fault-tolerance and state-management
- State migration
  - Hard limitation: Aggregated buckets in state snapshots cannot be migrated
- Sophisticated testing

**How to expose multi-windows and dynamic-windows to users?**

# Wrap-Up

**Scotty Features:**

- stream slicing
- pre-aggregation
- aggregate sharing
- out-of-order processing
- session window support
- multi-window support
- runtime-dynamic window support

**Let's bring it to production!**

JIRA: [FLINK-7001]

This talk is supported by the European Union Horizon 2020 Projects Proteus (687691), Streamline (688191), SAGE (671500), and E2Data (780245) and by the German Ministry for Education and Research as Berlin Big Data Center (01IS14013A) and Software Campus (01IS12056).