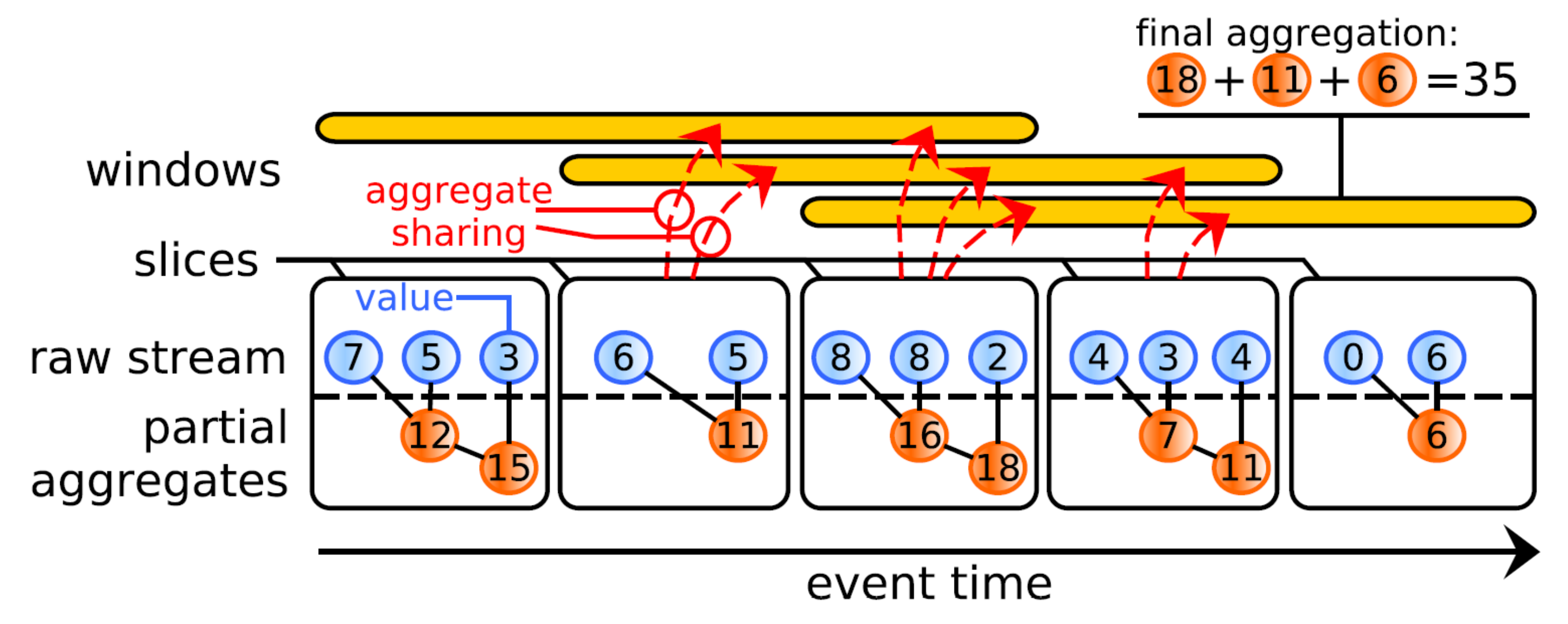
34th IEEE International Conference on Data Engineering (ICDE), April 16-19, 2018, Paris

Scotty: Efficient Window Aggregation for Out-of-Order Stream Processing

Jonas Traub jonas.traub@tu-berlin.de
Philipp M. Grulich philipp.grulich@campus.tu-berlin.de
Alejandro Rodríguez Cuéllar alejandro.rodriguez88@gmail.com
Sebastian Breß sebastian.bress@dfki.de
Asterios Katsifodimos a.katsifodimos@tudelft.nl
Tilmann Rabl rabl@tu-berlin.de
Volker Markl volker.markl@tu-berlin.de

Abstract and Contributions



Stream processing applications compute aggregates for overlapping windows, which causes redundant computations. Scotty splits streams into non-overlapping slices and computes partial aggregates per slice. We share partial aggregates among queries and windows to prevent redundant computations and to increase throughput.

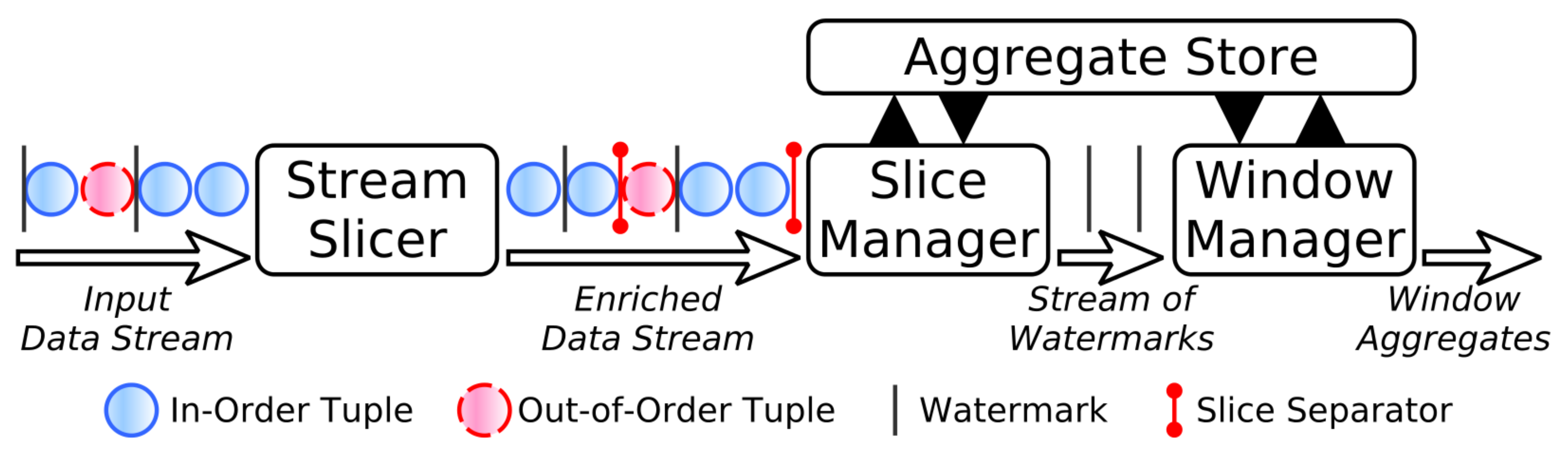
We make the following contributions:

1) We present Scotty, an operator for efficient streaming window aggregation which enables stream slicing, pre-aggregation, and aggregate sharing for out-of-order streams and session windows.

2) We show that stream slicing, pre-aggregation, and aggregate sharing are beneficial for concurrent session windows.

3) We design a Slice Manager which retains the minimum number of stream slices when processing out-of-order streams.

Scotty Architecture Overview



Stream Slicer:

- **Based on In-Order Tuples**: Split the stream into non-overlapping slices, for which we compute partial aggregates.

Slice Manager:

- **Based on Out-of-Order Tuples**: Update past slices when out-of-order tuples arrive (adding slices, fusing slices, changing the start and end timestamps of slices, and updating partial aggregates).
- Notify the Aggregate Store about the start and end of slices.

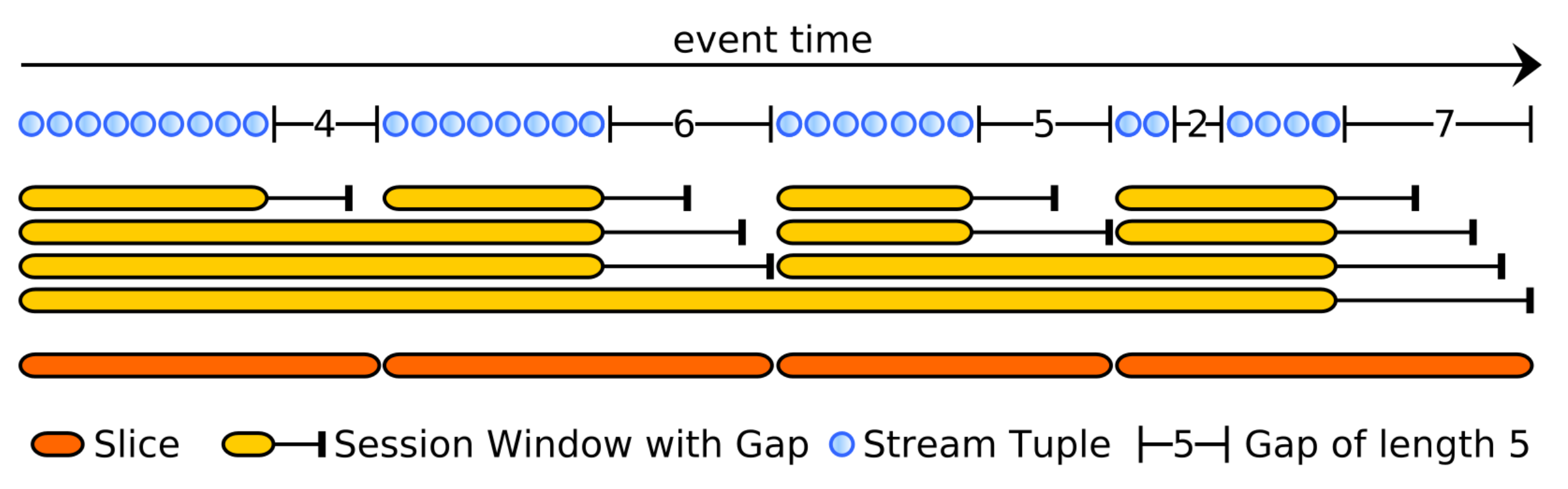
Aggregate Store:

- Compute aggregates and store partial aggregates for slices.
- We implement and evaluate *lazy aggregation* (one partial aggregate per slice) and *eager aggregation* (aggregate tree on top of slices). [see: *Cutty: Aggregate Sharing for User-defined Windows, Carbone et al., 2016*]

Window Manager:

- Combine partial aggregates to final aggregates for windows and output final aggregates when windows end.

Stream Slicing and Aggregate Sharing for Session Windows



Stream Slicing Example:
Concurrent Session Windows with gaps 3,5,6, and 7

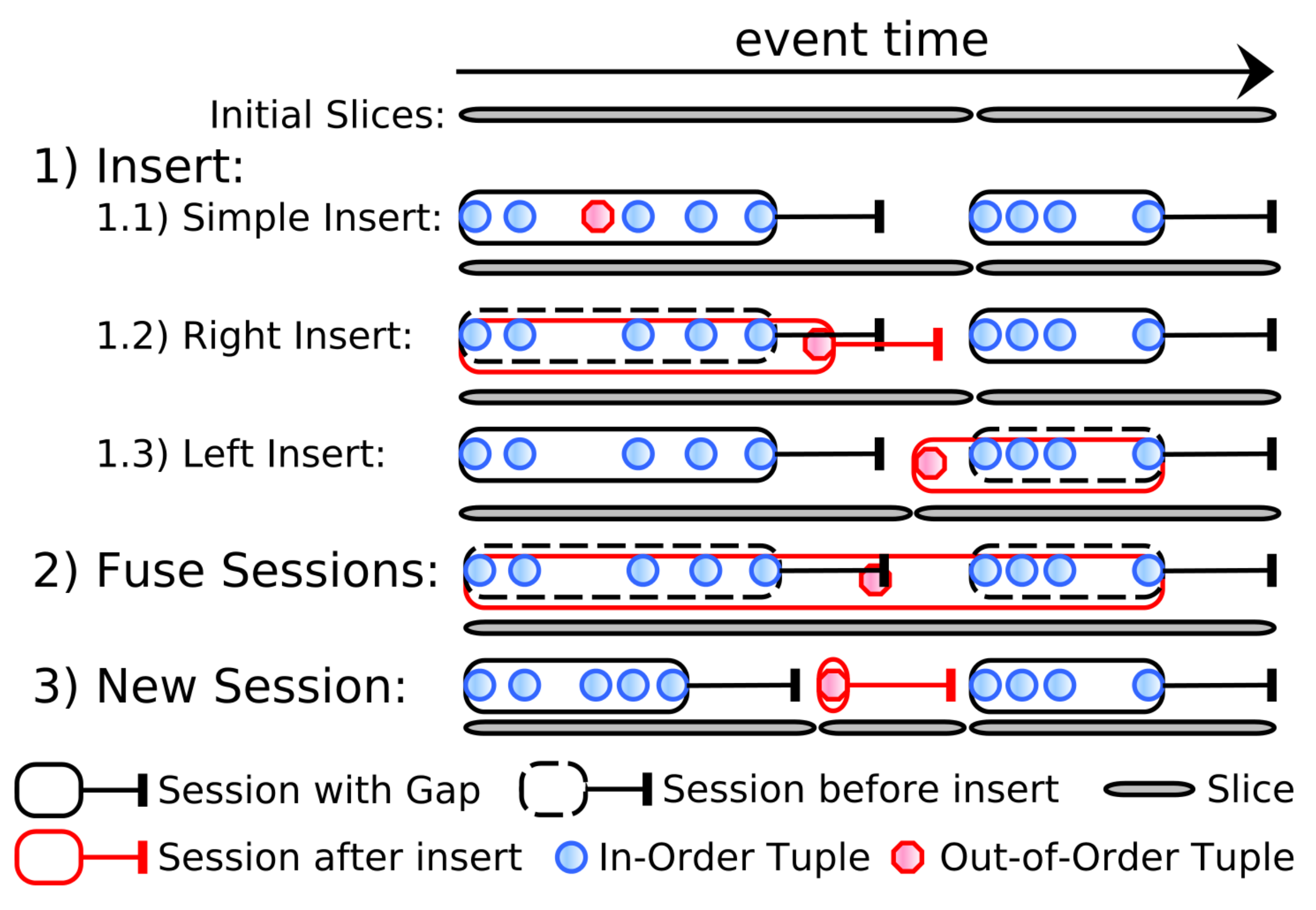
Key Observations:

1) Multiple session windows with different gaps share partial aggregates.

2) Session windows can share aggregates with concurrent sliding and tumbling windows.

3) A slice that covers a session and a gap is logically equivalent to a slice that covers the session only because gaps contain no tuples per definition.

4) The slicing logic depends solely on the session window with the smallest gap. Session windows with larger minimum gaps can use the same slices.



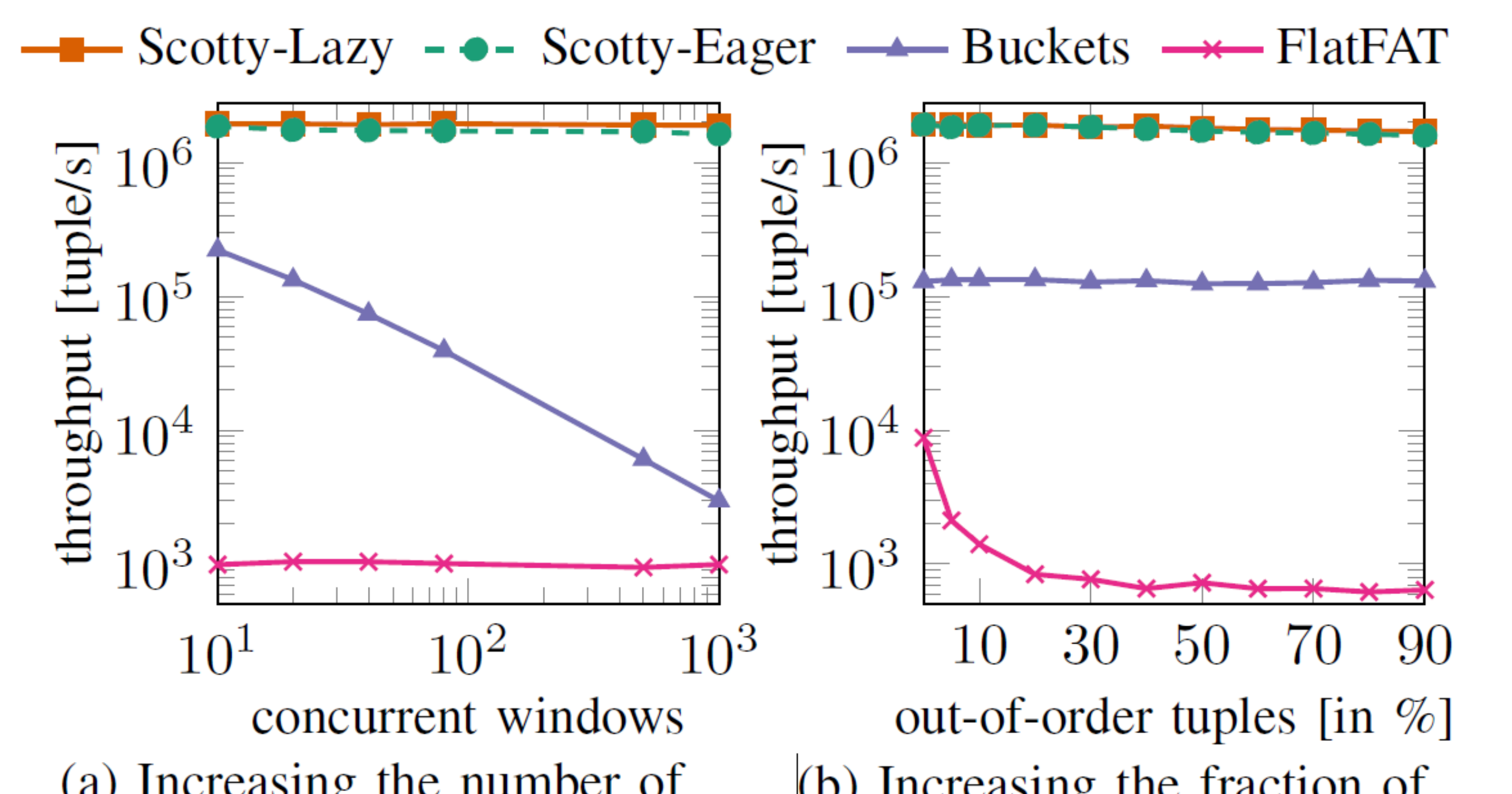
Out-of-Order Processing and Sessions:

Scotty's Slice Manager always retains the minimum number of slices when processing out-of-order tuples.

Each out-of-order tuple leads to one of the updates shown in the Figure on the left.

There is no need to store individual tuples, because Scotty splits slices in gaps only. Thus, it is sufficient to store one partial aggregate per slice. This reduces the memory footprint.

Experiments Summary



(a) Increasing the number of concurrent windows. (b) Increasing the fraction of out-of-order tuples.

(a) Concurrent Windows (i.e., Queries):

- Scotty achieves an order of magnitude higher throughput than alternative techniques which do not use stream slicing.
- Scotty scales to large numbers of concurrent windows with almost constant throughput.

(b) Out-of-Order Processing:

- Scotty processes out-of-order tuples efficiently because there are just a few hundred slices. This makes it fast to find the correct slices for out-of-order tuples.
- Scotty scales to high fractions of out-of-order tuples with an almost constant throughput.

Technische Universität Berlin

German Research Center for Artificial Intelligence

Deutsche Forschungsgemeinschaft

Bundesministerium für Bildung und Forschung

This work was supported by the EU project Proteus (687691), DFG Stratosphere (606902), and the German Ministry for Education and Research as BBDC (01IS14013A) and Software Campus (01IS12056).