22nd International Conference on Extending Database Technology (EDBT), March 26-29, 2019, Lisbon, Portugal

Efficient Window Aggregation with General Stream Slicing

Jonas Traub
jonas.traub@tu-berlin.de

Philipp M. Grulich
philipp.grulich@campus.tu-berlin.de

Alejandro Rodríguez Cuéllar
alejandro.rodriguez88@gmail.com

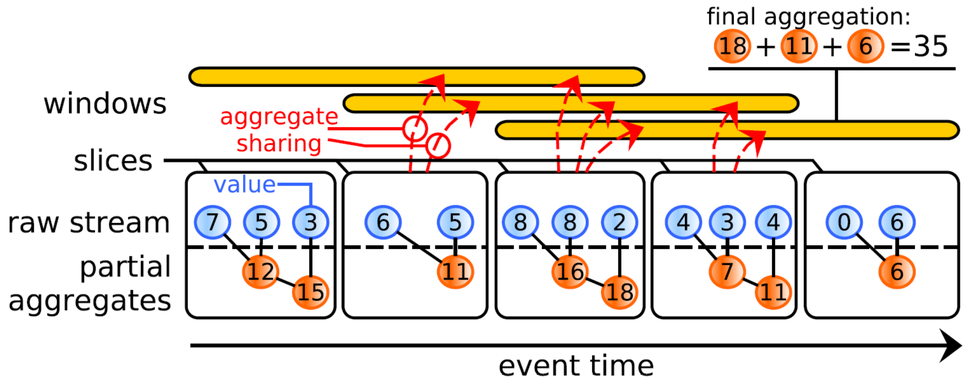
Sebastian Breß
sebastian.bress@dfki.de

Asterios Katsifodimos
a.katsifodimos@tudelft.nl

Tilmann Rabl
rabl@tu-berlin.de

Volker Markl
volker.markl@tu-berlin.de

Abstract and Contributions

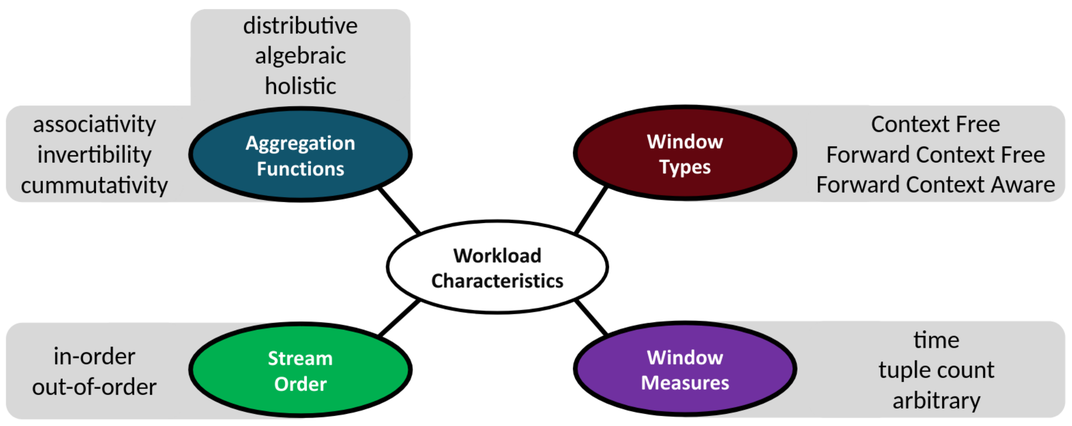


General stream slicing automatically adapts to workload characteristics to improve window aggregation performance without sacrificing general applicability.

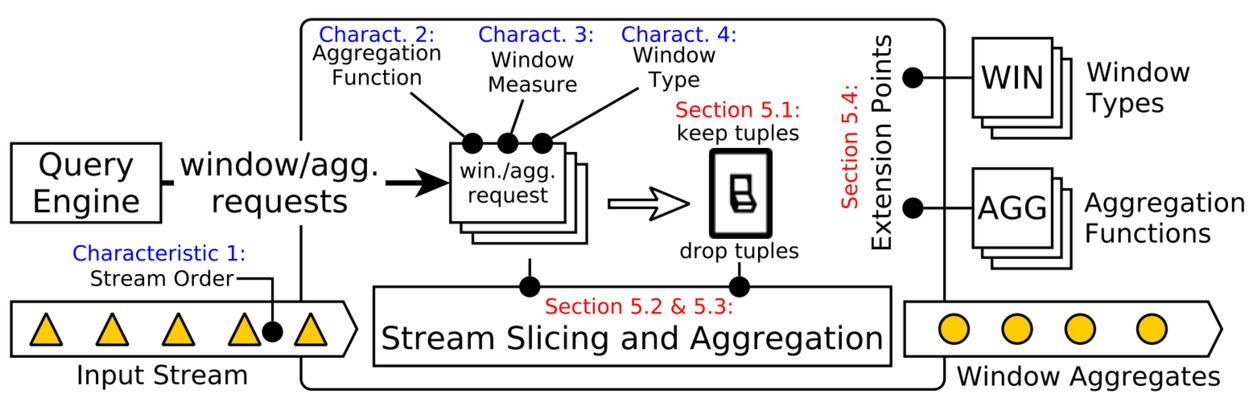
We make the following contributions:

1. We identify the workload characteristics which impact the applicability and performance of window aggregation techniques.
2. We contribute general stream slicing, a generally applicable and highly efficient solution for streaming window aggregation.
3. We analyze the implications of workload characteristics and show that stream slicing is generally applicable while offering better performance than existing approaches

Workload Characteristics

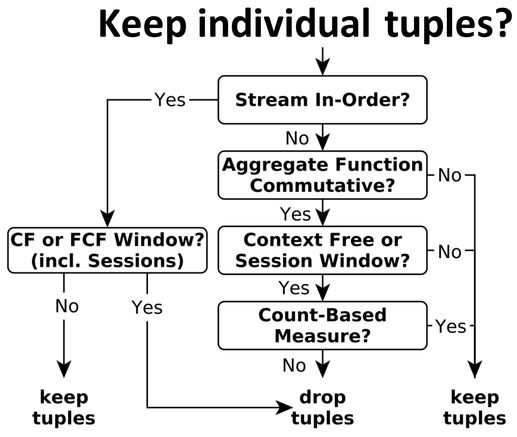


Architecture of General Stream Slicing



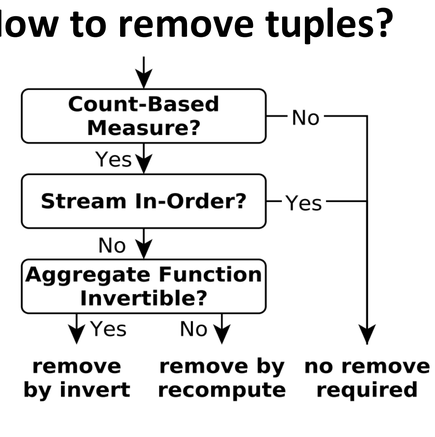
Adapting to Workloads Characteristics

Keep individual tuples?

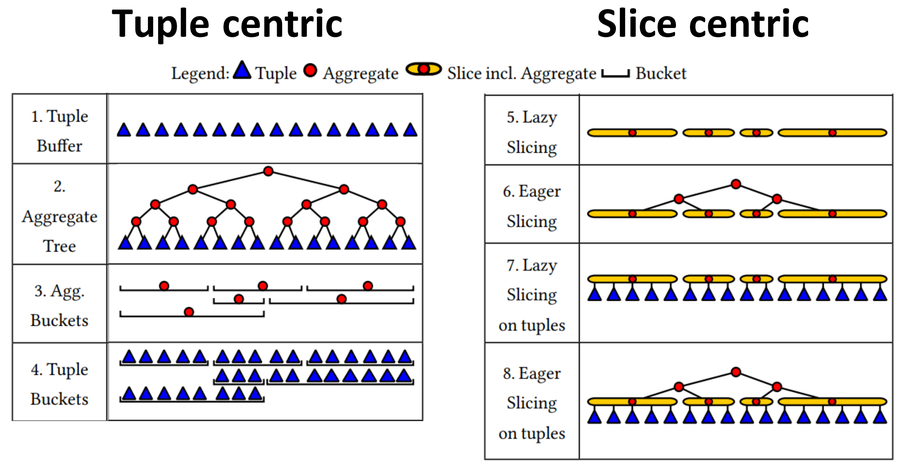


Are splits required?

How to remove tuples?

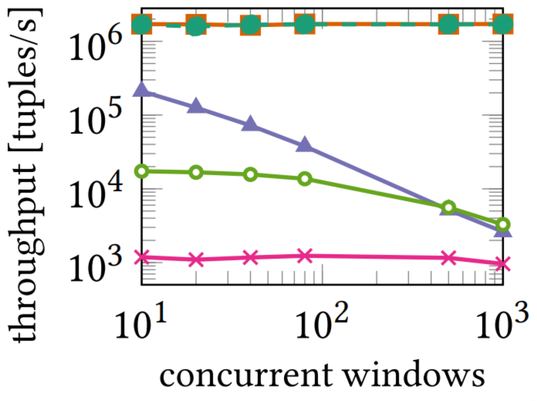


Aggregation Techniques

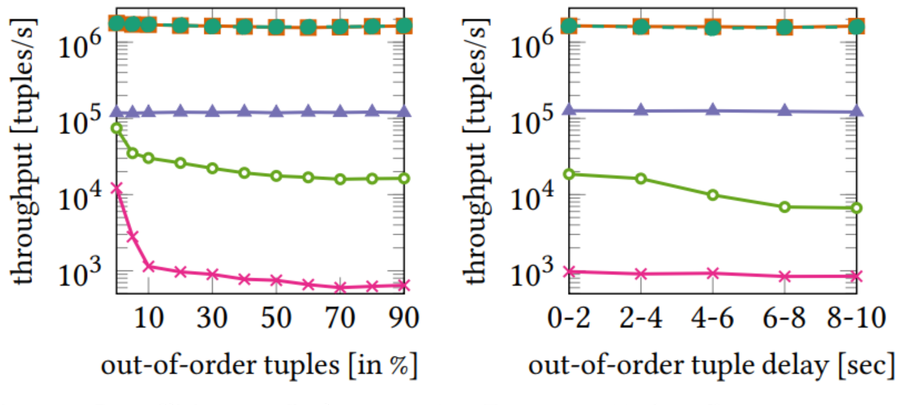


Performance Evaluation

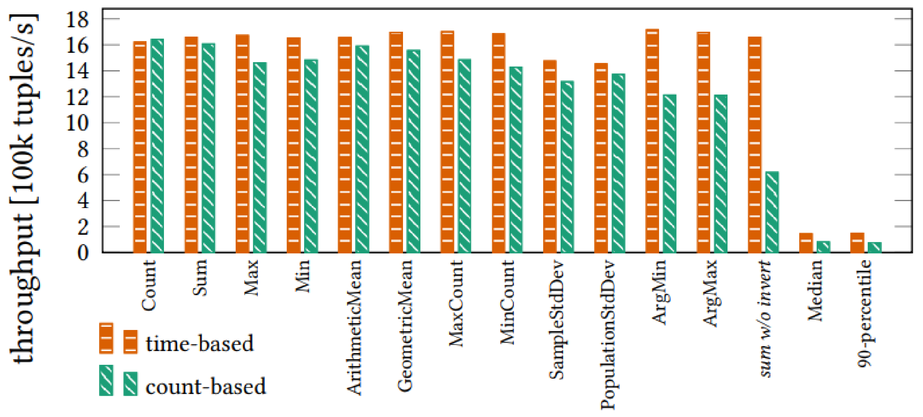
Scalability (20% out-of-order)



Impact of Out-Of-Order Tuples



Impact of Aggregation Functions (20% out-of-order)



Lazy Slicing — Eager Slicing — Buckets — Agg. Tree — Tuple Buffer

Key findings

- General slicing outperform alternative concepts with respect to throughput and scales to large numbers of concurrent windows.
- Stream slicing and Buckets scale with constant throughput to large fractions of out-of-order tuples and are robust against high delays of these tuples.
- On time-based windows, stream slicing performs diverse distributive and algebraic aggregations with similarly high throughputs. Considering count-based windows and out-of-order tuples, invertible aggregations lead to higher throughputs than not invertible ones.

Technische Universität Berlin

German Research Center for Artificial Intelligence

Open Source Repository:
tu-berlin-dima.github.io/scotty-window-processor