26/03/26

Mobility Trajectory Data Stream Processing Beyond the Cloud

Mariana Duarte, Dwi P.A. Nugroho, Georges Tod,
Evert Bevernage, Pieter Moelans, Elias Saerens,
Esteban Zimányi, Mahmoud Sakr, Steffen Zeuch

mariana.machado.garcez.duarte@ulb.be

Outline

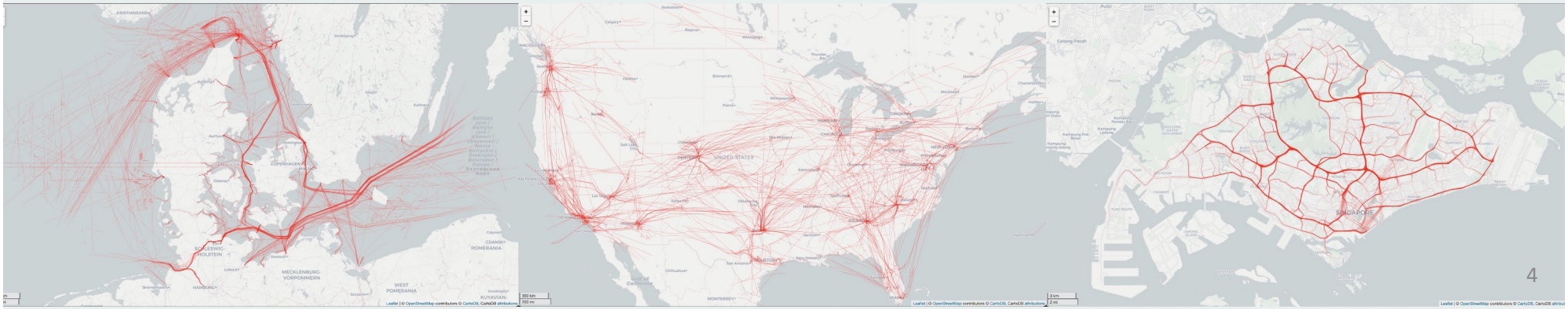
1. Introduction & Motivation
2. System Components: MEOS + NebulaStream
3. Windowed Trajectories
4. SNCB Use Case & Queries
5. Evaluation Results
6. Conclusion & Future Work

Introduction and Motivation

Motivation: The Explosion of Mobility Data

Why Mobility Stream Processing?

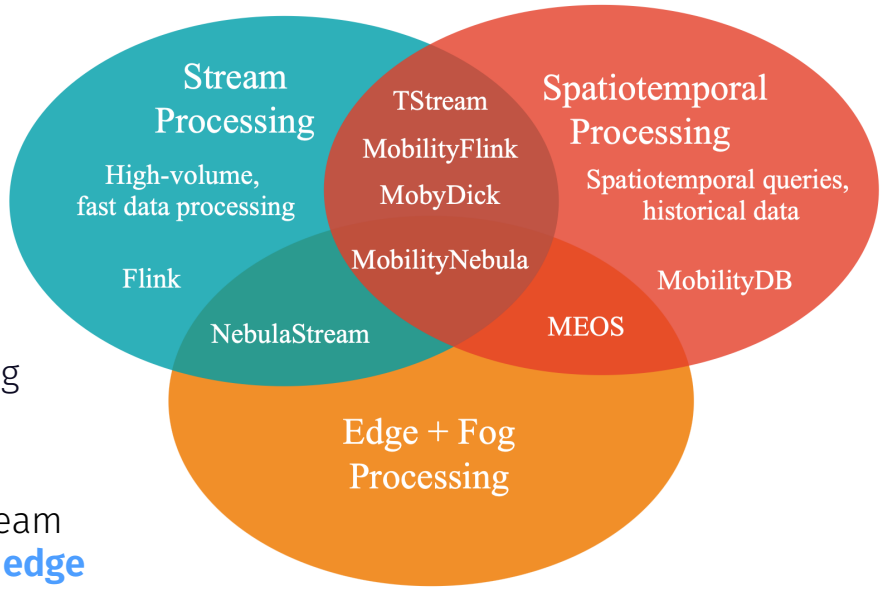
- IoT sensors in moving objects generate **high-volumes** of mobility data
- Require **efficient spatial analytics** and real-time processing
- Connectivity is limited and/or unstable (in trains, maritime, rural)
- **Local processing** enables fast decision-making
- Reduces latency and cloud dependency
- Applications: **routing, emergency response, maintenance prediction**



Research Gap

Current Systems Are Not Enough

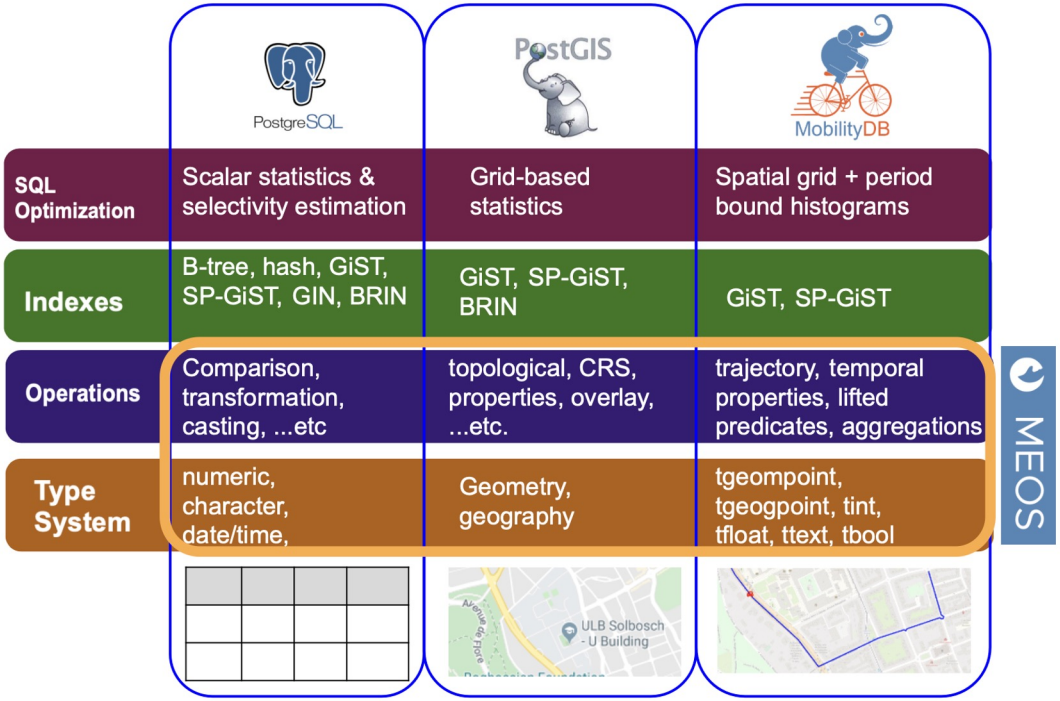
- Stream processing systems **lack** spatiotemporal capabilities
- Spatiotemporal databases manipulate **historical** data
- Edge and fog processing do not have **spatiotemporal awareness** and stream processing management
- MobilityNebula bridges this gap by integrating stream processing and **spatiotemporal awareness at the edge**



System Components

MEOS: Mobility Engine Open Source

- MEOS [1]: mobility management
- **C library** for spatiotemporal data
- Specialized data types: trajectories, bounding boxes, moving geometries
- **Extensive library** with more than 1300 functions
- Deployable at the **edge** and/or **cloud**
- MEOS computation is decoupled from storage



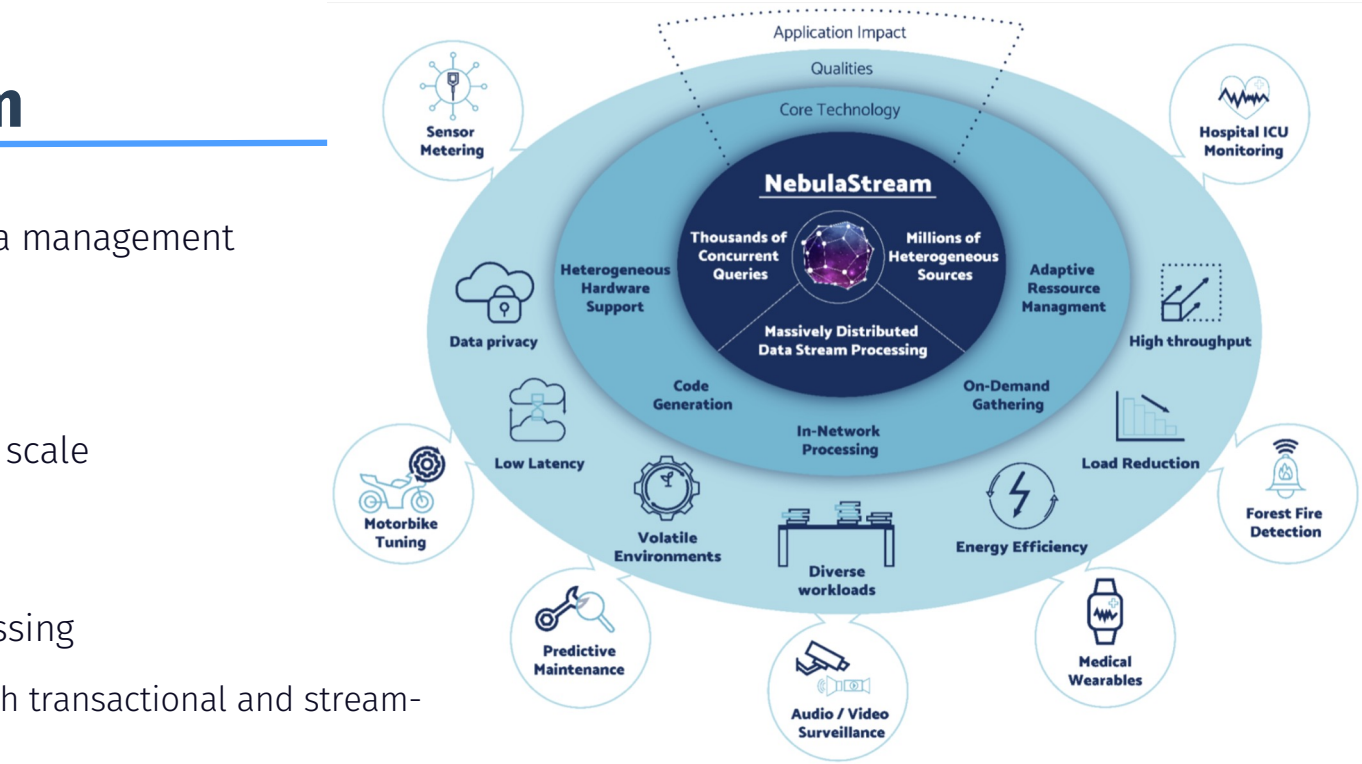
[1] Zimányi, Esteban, Mariana MG Duarte, and Víctor Diví. "MEOS: An Open Source Library for Mobility Data Management." International Conference on Extending Database Technology. 2024.

NebulaStream

- General purpose stream data management system for the **IoT** [2]

- Geo-distributed, dynamic
- Heterogeneous and massive scale
- Written in **C++**

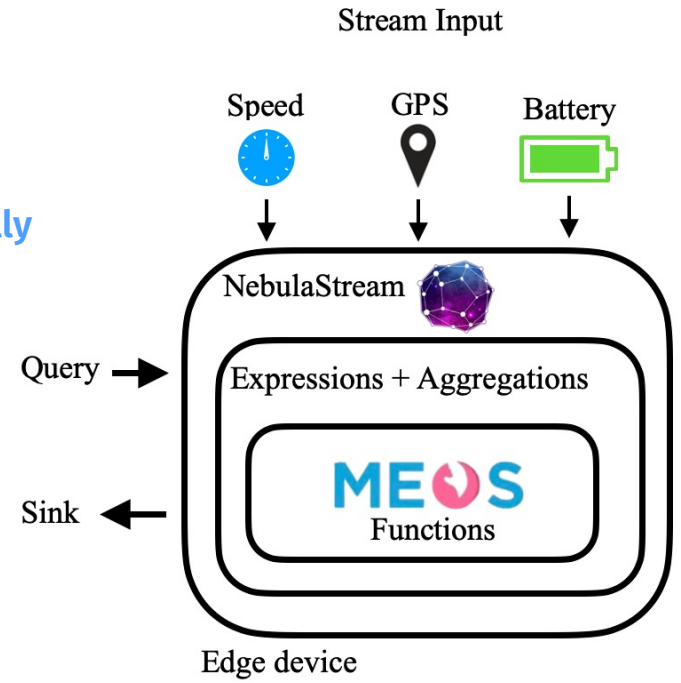
- **Transactional** Stream Processing
  - Merges advantages of both transactional and stream-oriented guarantees
- Ideal system for mobility workloads



[2] Zeuch et al. (2022). NebulaStream: Data Management for the Internet of Things. Datenbank-Spektrum. 22. 10.1007/s13222-022-00415-0.

MobilityNebula

- Integrates MEOS into NebulaStream
- Sensors send data to **edge processor**
- Edge device runs MobilityNebula, processes data **locally**
- Processed data sent via sink to storage (Cloud, MQTT, MobilityDB)



Edge Execution

Deploying and running MobilityNebula programs on edge devices

Raspberry Pi 4 (on battery)

NVIDIA Jetson AGX Orin

Window-Aware Trajectory Type System

1

Abstract Model

moving(SPATIAL) → TEMPORAL — Objects whose positions change over time

2

Discrete Model

tinstant, tsequence, tsequenceset — Sequence representation, each observation stored once

3

Streaming Extension

stinstant, stsequence, stsequenceset — Bind TEMPORAL × WINDOW → STEMPORAL

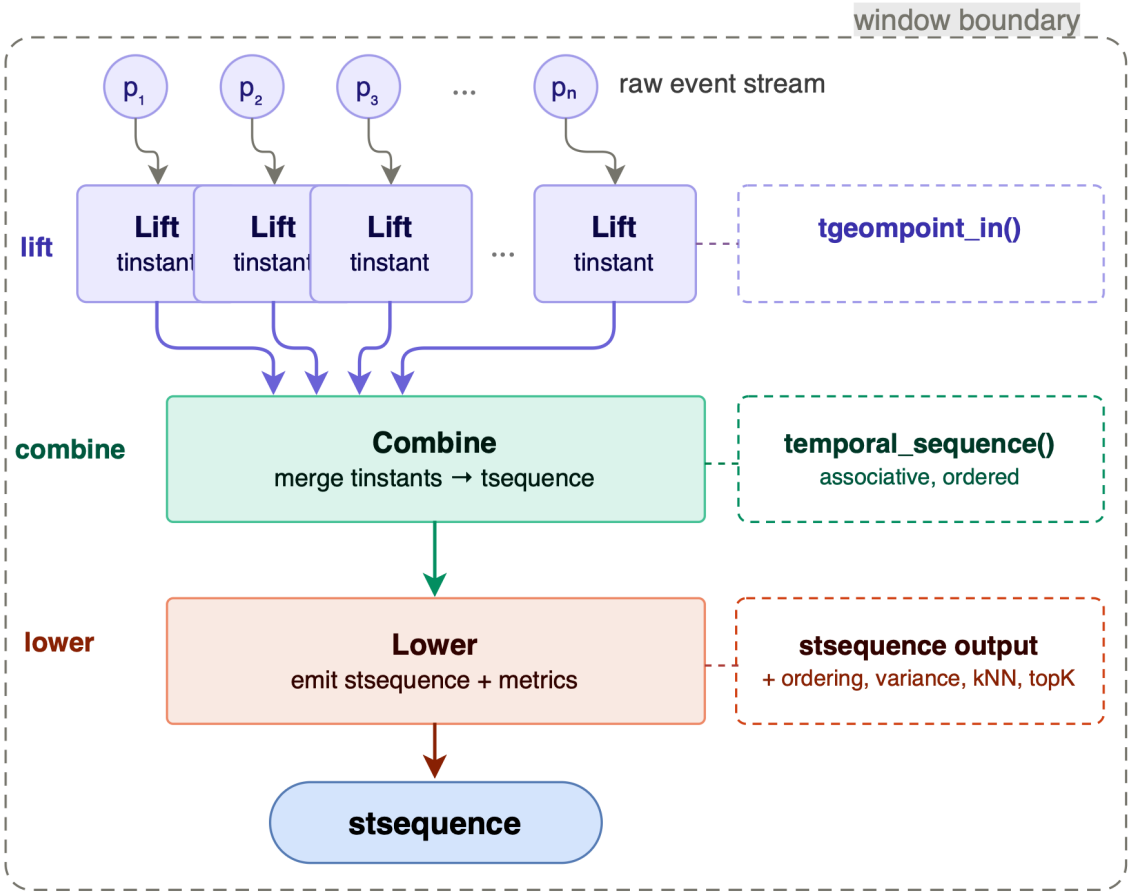
Key: *each sequence is coupled with its window — a single value for streaming analytics*

Aggregation as Trajectory Construction

Lift-Combine-Lower Pattern:

Lift	Raw tuple → instants
Combine	Merge tinstants into tsequence
Lower	Output stsequence + extract metrics

Event-time watermarks *drive window closure. Late arrivals before watermark are added. After watermark, the window is closed.*



Edge-Executable Spatiotemporal Operators

Function	Description
`edwithin_tgeo_geo`	Proximity check: is temporal point within distance of geometry?
`eintersects_tgeo_geo`	Intersection test: does temporal point intersect geometry?
`tgeo_at_stbox`	Containment: is temporal point inside a spatiotemporal box?
`nad_tgeo_stbox`	Nearest approach distance to a spatiotemporal box
`nearest_approach_distance`	Minimum distance between two temporal points over time
`temporal_sequence`	Constructs a windowed trajectory from GPS points
`temporal_ext_kalman_filter`	Trajectory smoothing with Extended Kalman Filter

All operators are composable with windowing, grouping, map, and filter.

SNCB Use Case

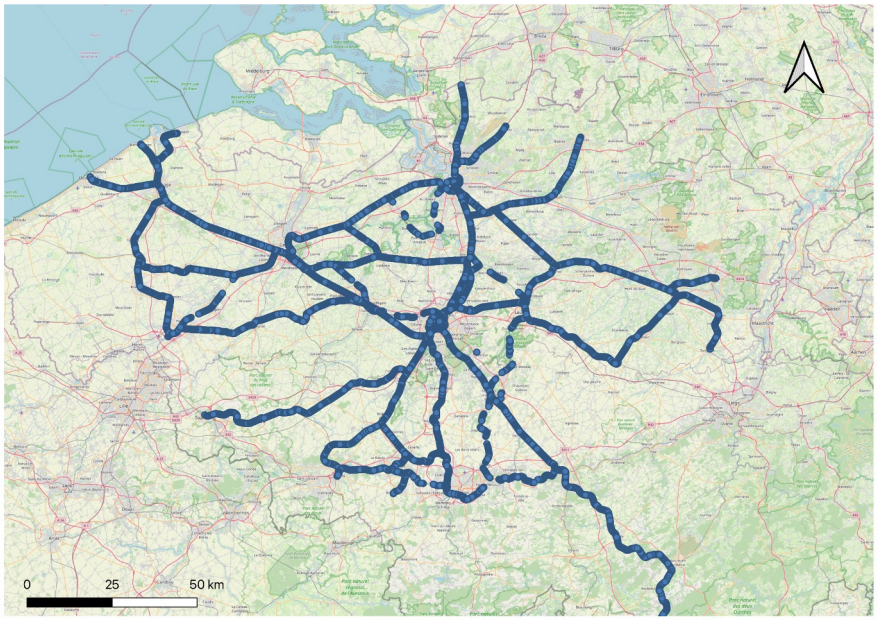
SNCB Trains (1/2)

- Partnered with the National Railway Company of Belgium
- M6 trains (2002)
- Goal: real-time **analytics onboard** Belgian trains
  - Predictive maintenance
  - Faster emergency response

SNCB Trains (2/2)

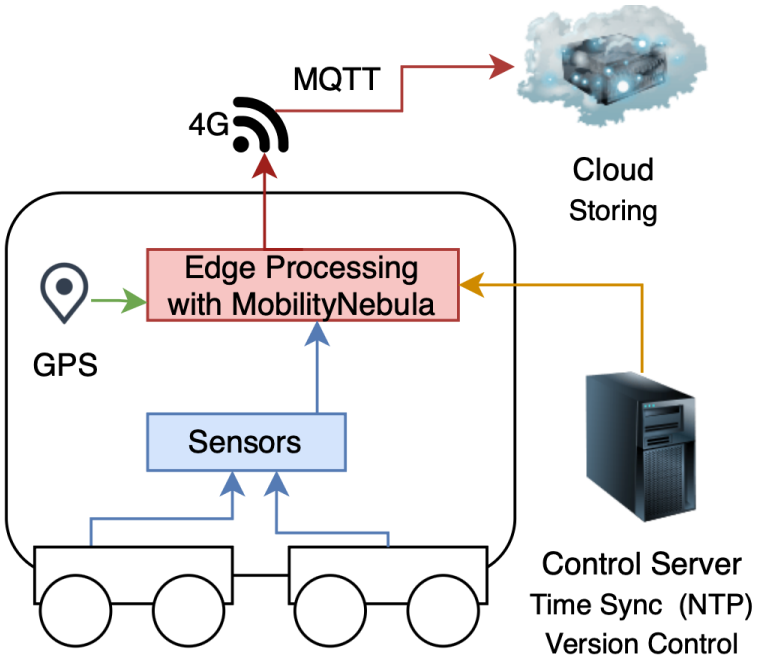
- Trains did not have sensors originally
- SNCB installed **GPS**, **brake pressure** sensors, and **battery** measurement
- Sensors send data to a nROK device (running **Ubuntu**)

nROK edge device



Data coverage

Real-World Deployment



Design Principles

- Safety decisions processed locally on-train
- Cloud used only for storage & offline analysis
- MQTT alerts over 4G when connectivity available
- Operates during intermittent connectivity
- Replaces legacy PLC-based diagnostics
- Remote query deployment from HQ

Queries for Railway Monitoring

Q1 **High-Risk Zone Proximity**

edwithin_tgeo_geo + tumbling

Q2 **Brake System Monitoring**

eintersects + variance agg

Q3 **Trajectory Creation**

temporal_sequence + sliding

Q4 **Trajectory in Restricted Space**

tgeo_at_stbox + trajectory

Q5 **High-Speed Alert + Trajectory**

geofence + speed aggregation

Q6 **Positional Divergence**

join + nearest_approach_dist

Q7 **Global Closest Pairs (Top-k)**

join + topK selection

Q8 **Trajectory Smoothing**

Extended Kalman Filter [3]

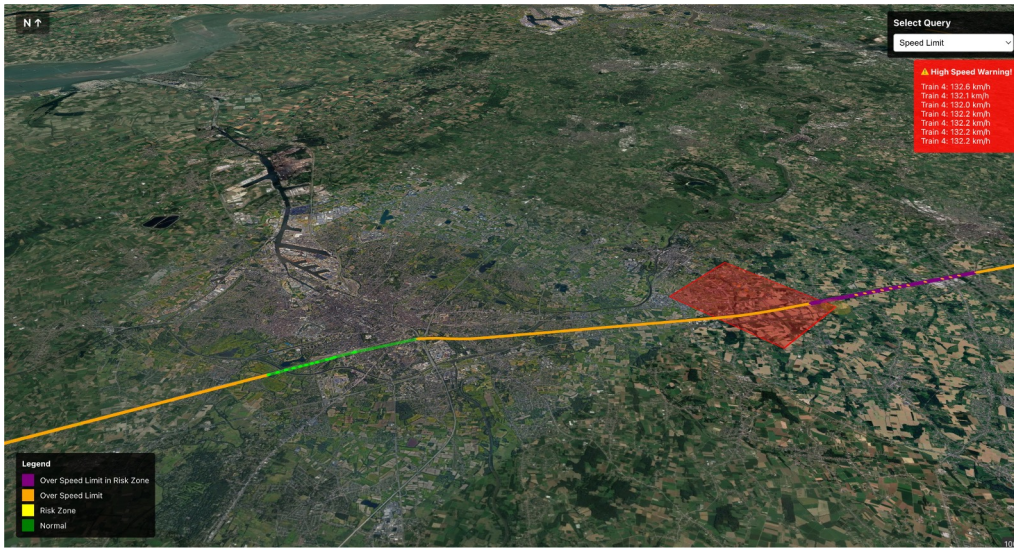
Q9 **Windowed Per-Device kNN**

join + knn_agg (k=3)

Queries cover: geofencing, trajectory construction, brake monitoring, joins, kNN, trajectory smoothing

[3] https://github.com/simondlevy/TinyEKF/tree/master.

Query Visualization



- Brake monitoring
- High-risk zone proximity

[3] Mariana M Garcez Duarte et al. (2025). Mobility Stream Processing on NebulaStream and MEOS. SIGMOD.

Evaluation Results

Evaluation Setup

Baselines	TStream (Flink), MobyDick (Flink), GeoEKuiper
Workload	20k events/s TCP stream, 9 queries, 30s runs
Metrics	Throughput, p95 latency, CPU, peak memory, success rate
Memory caps	512 MB, 2 GB, 8 GB
CPU caps	1, 2, 3, 4 vCPUs
Profiles	Raspberry Pi 5 (4 vCPU/8 GB), nROK (2 vCPU/8 GB)

Query Support Across Systems

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
MobilityNebula	✓	✓	✓	✓	✓	✓	✓	✓	✓
MobyDick	✓	✓	✓	✓	✓	—	—	—	—
TStream	✓	✓	✓	✓	✓	—	✓	—	✓
GeoEKuiper	✓	✓	✓	✓	✓	✓	✓	—	✓

Edge-Constrained Performance Across Systems

20k
events/s
Sustained throughput at 2+ vCPUs

148–545
MB
Peak memory across all queries

91.9%
Success rate (vs 54–73% baselines)

≤1.1s
p95 latency at 8GB/2vCPU

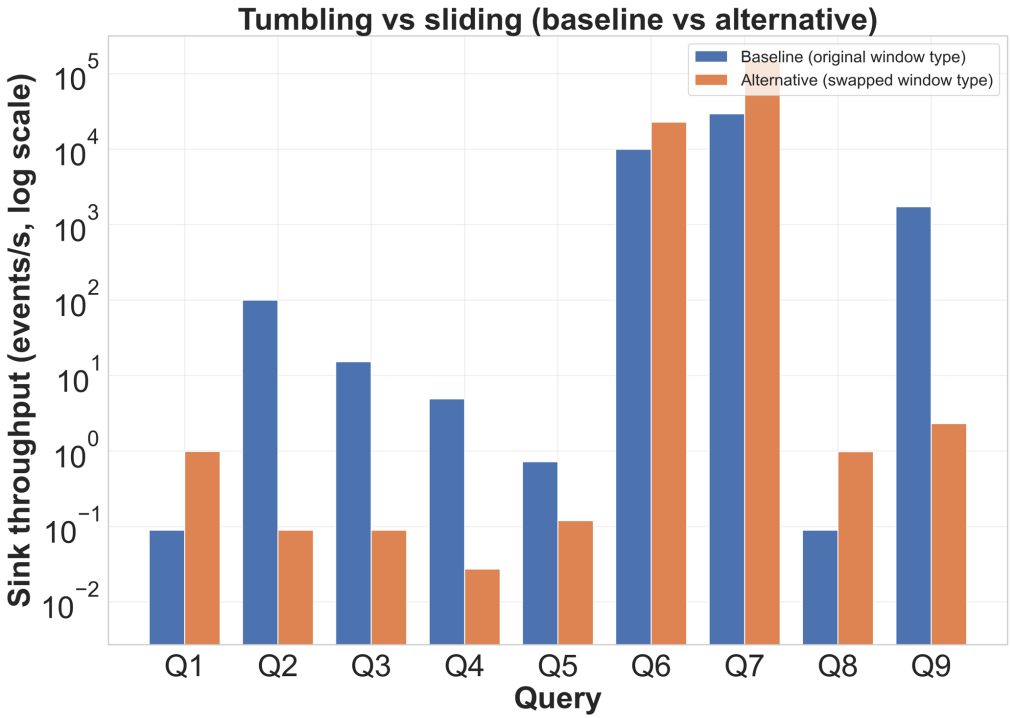
Key Findings

- **MobilityNebula** scales throughput with vCPUs (17k → 20k from 1 to 2+ vCPUs); baselines are memory-bound
- **MobyDick** peaks at 1.5 GB memory, 207% CPU on trajectory queries — not viable on edge
- **TStream** non-viable at 512 MB (oom_kill on most queries); needs ≥ 2 GB of memory
- **GeoEKuiper**: stable throughput but high tail latencies and memory usage (1.5 GB on Q2)

Parameter Sensitivity in MobilityNebula

Parameter Sensitivity

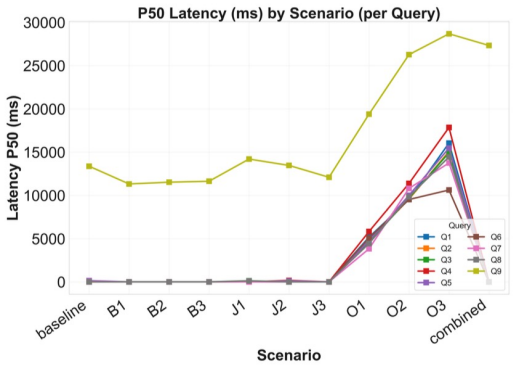
- **Slide interval** is the main driver: reducing from 1s to 10ms increases sink throughput 100×
- **Q3 at 10ms slide**: throughput drops to 4k events/s, memory rises to 2.7 GB
- **Q3 at 1s slide**: throughput recovers to 20k events/s, memory 611 MB
- **Join workloads** (Q6, Q7) most sensitive: throughput degrades under sliding
- **Tumbling → sliding** conversion increases overlap costs proportionally



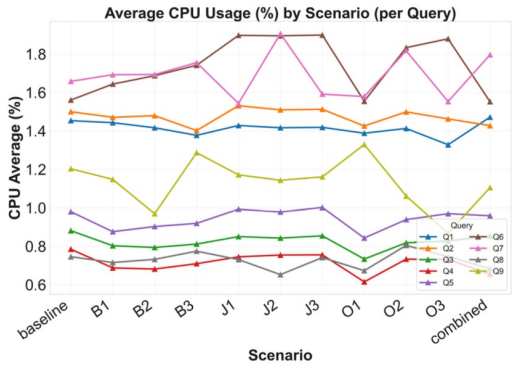
Network Volatility in MobilityNebula

Network Volatility Stress

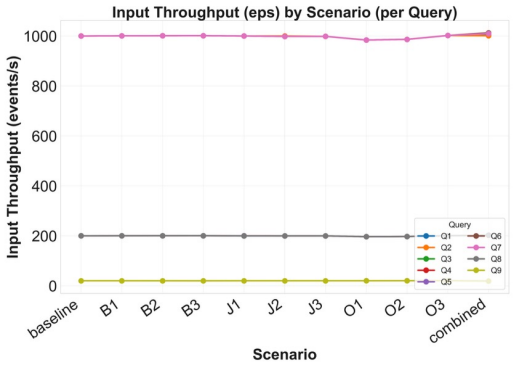
- **All 9 queries pass** all scenarios (jitter, bursts, outages, combined)
- **CPU: 1–2%,** memory: 160–220 MB (stable)
- **Jitter** (±20–200ms): minimal latency impact
- **Bursts** (on/off cycles): moderate latency increase
- **Outages** (10–30s): most disruptive — watermark stalls
- **Combined** (J2+B2+O1): better than worst outage alone



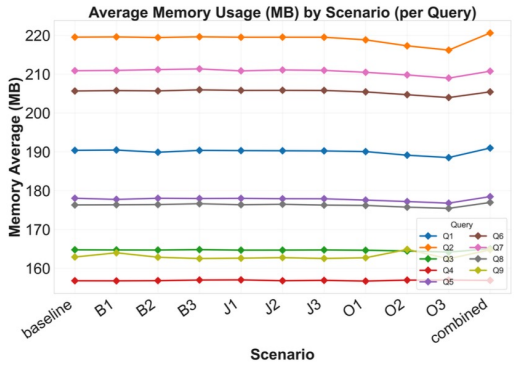
(a) p50 latency



(b) CPU usage



(c) Throughput



(d) Average memory usage

Conclusion & Future Work (1/2)

What We Showed

- MobilityNebula enables trajectory-based streaming analytics on edge devices
- Created window-aware type system (stinstant, stsequence, stsequenceset)
- Used lift-combine-lower aggregation for incremental trajectory construction
- Spatiotemporal operator integration
- Real World deployment in SNCB trains
- 91.9% success rate vs 54–73% for baselines

Conclusion & Future Work (2/2)

Future Directions

- In-memory spatial indexes for faster streaming queries
- GPU acceleration for compute-heavy operators (kNN, EKF)
- Distributed placement optimizations for spatiotemporal operators
- ~~Transport-level disconnect behavior: alarm continuity, buffer overflow, recovery~~


Thank you
obrigada
github.com/MobilityDB/MobilityNebula
Funded by EU Horizon Europe — MobiSpaces (Grant No. 101070279)
mariana.machado.garcez.duarte@ulb.be
Atomium