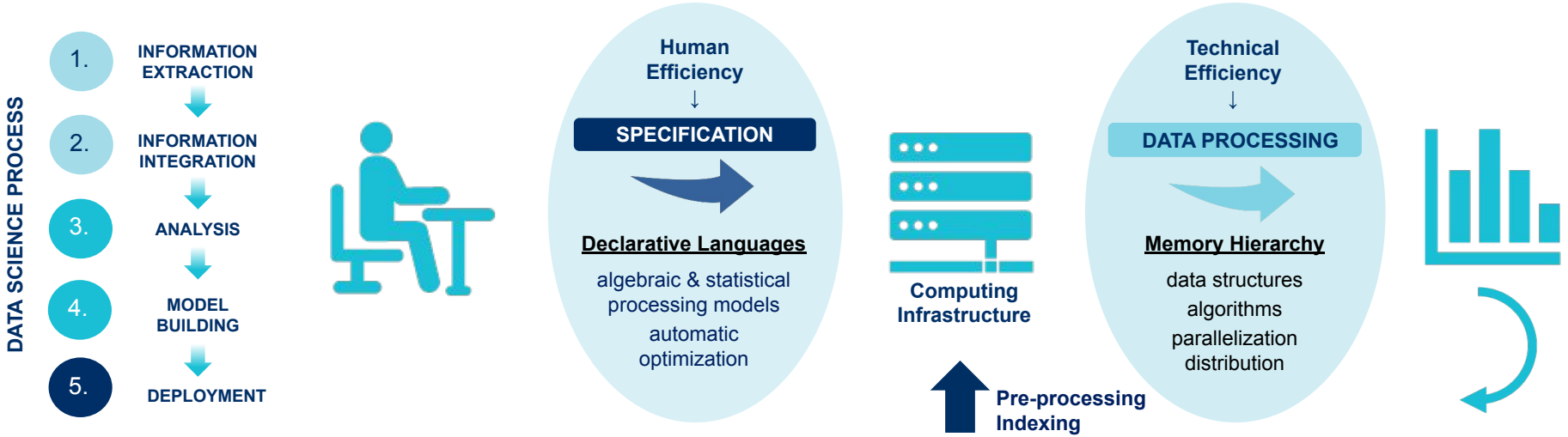
NebulaStream: Data Stream Processing for the Sensor-Edge-Cloud Continuum

ICDE 2026 Keynote · **Volker Markl** · TU Berlin / BIFOLD & DFKI

Agenda

1. **Background: Stream Data Management**
2. Recap: Apache Flink
3. NebulaStream – Vision & History
4. Current Applications
5. Design & Architecture
6. Open Source Status & Roadmap
7. Research Topics
8. Summary & How to Get Involved

Data Management Systems Research

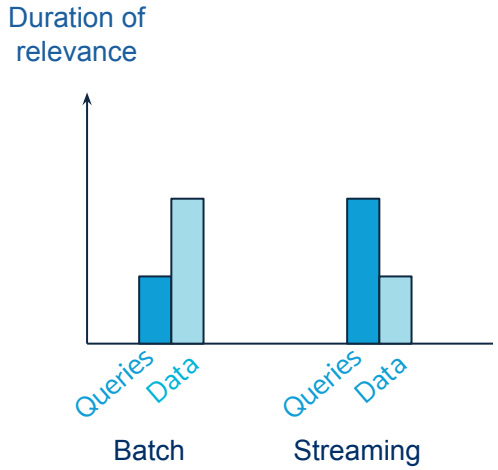


Goal of Data Management Systems Research:
Scaling the data science process and its deployment w.r.t. human & technical efficiency

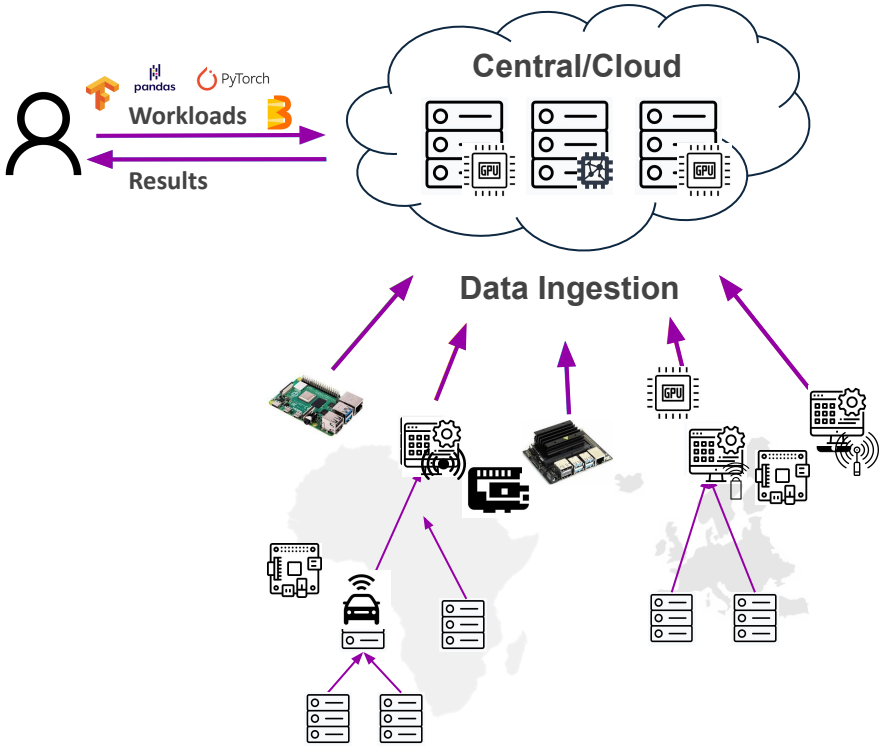
Draws on Results from:
computer architecture, distributed systems, compilers, statistics, and machine learning

Stream Processing (vs. Batch Processing)

- **Batch:** Static data (at rest), dynamic queries
  - historical data analysis
- **Stream:** Dynamic data (in motion), static queries
  - real-time data processing



The “Osmotic” Sensor-Edge-Cloud Continuum



- heterogeneous compute resources
- sensor-driven data ingestion
- hierarchically distributed topology
- geo-distribution
- moving devices

Cloud vs. “Osmotic” Sensor-Edge-Cloud

Cloud

- parallel
- scale out
- homogeneous hardware
- data born in the cloud
- reliable interconnects
- powerful processing
- delayed stream analysis

“Osmotic” Sensor-Edge-Cloud

- massively distributed
- scale out and scale up
- heterogeneous hardware
- data born in the edge
- unreliable network
- resource-constrained processing
- real-time stream analysis and actuation

Agenda

1. Stream Data Management
2. **Recap: Apache Flink**
3. NebulaStream – Vision & History
4. Current Applications
5. Design & Architecture
6. Open Source Status & Roadmap
7. Research Topics
8. Summary & How to Get Involved

Apache Flink: Declarative Fast Data Stream Analytics in the Cloud

Human Efficiency

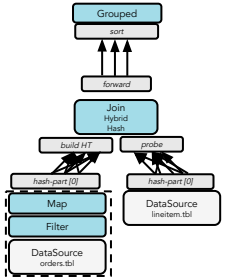
```
case class Path (from: Long, to: Long)
val tc = edges.iterate(10) {
  paths: DataSet[Path] =>
    val next = paths
      .join(edges)
      .where("to")
      .equalTo("from") {
        (path, edge) =>
          Path(path.from, edge.to)
      }
      .union(paths)
      .distinct()
    next
}
```

Declarative Data Analysis Program

Algebraic Model

Cost Model

Automatic Optimization, Parallelization & Hardware Adaption



Data Flow Graph

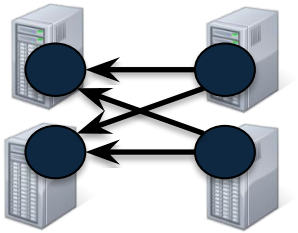
Specification Time

Memory Management

Out-of-Core Algorithms

Data Stream Processing

State Management



Worker

Distribution of Operators

Monitoring of the Operation

Fault Tolerance

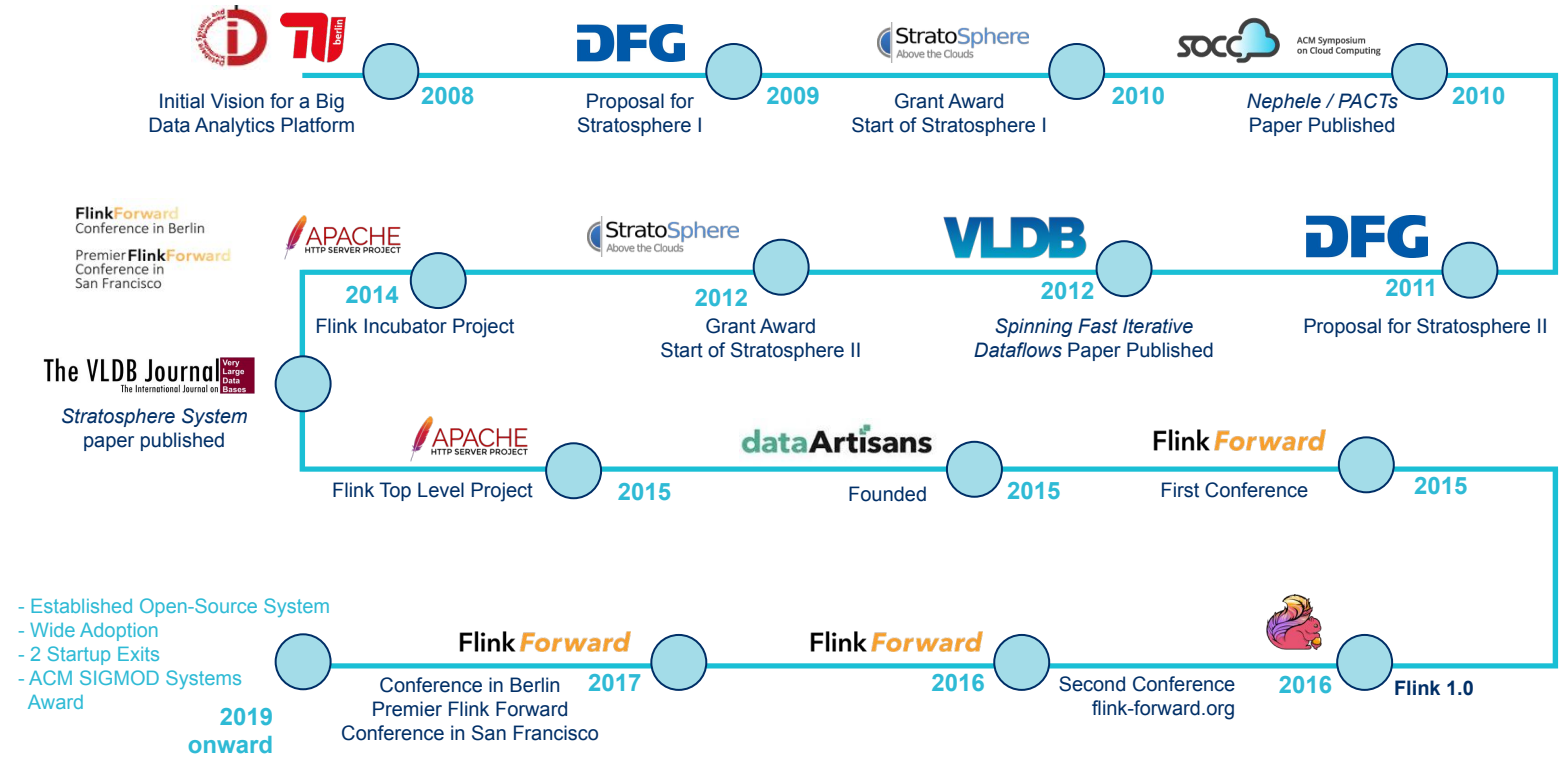
Resource Management

Runtime

Coordinator

Technical Efficiency

Innovation Pipeline
from Stratosphere to Flink

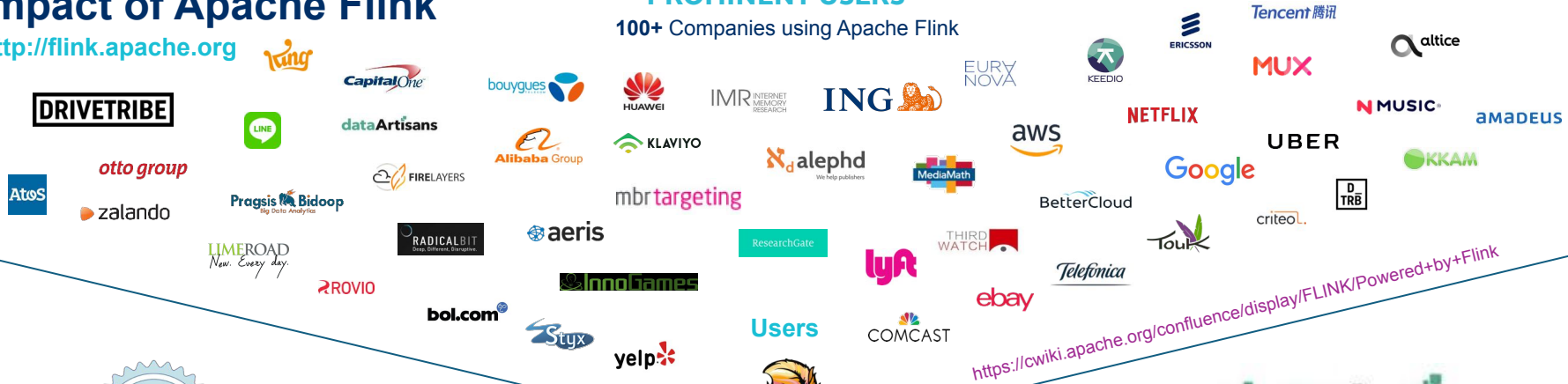


Impact of Apache Flink

http://flink.apache.org

PROMINENT USERS

100+ Companies using Apache Flink



https://cwiki.apache.org/confluence/display/FLINK/Powered+by+Flink

Users

Awards

ACM SIGMOD Systems Award 2023

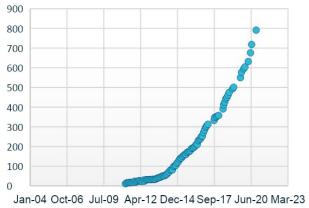
Startup Exits

dataArtisans

IMMEROK
IMMEROK
IMMEROK
IMMEROK

Community

OPEN-SOURCE CODE CONTRIBUTORS



> 1600

(as of Feb. 2026)

45,000+ Meetup Members Worldwide

60+ Meetup Groups Worldwide

20+ Countries that Regularly Hold Meetups

https://www.flink-forward.org/

Asia
Europe
North America

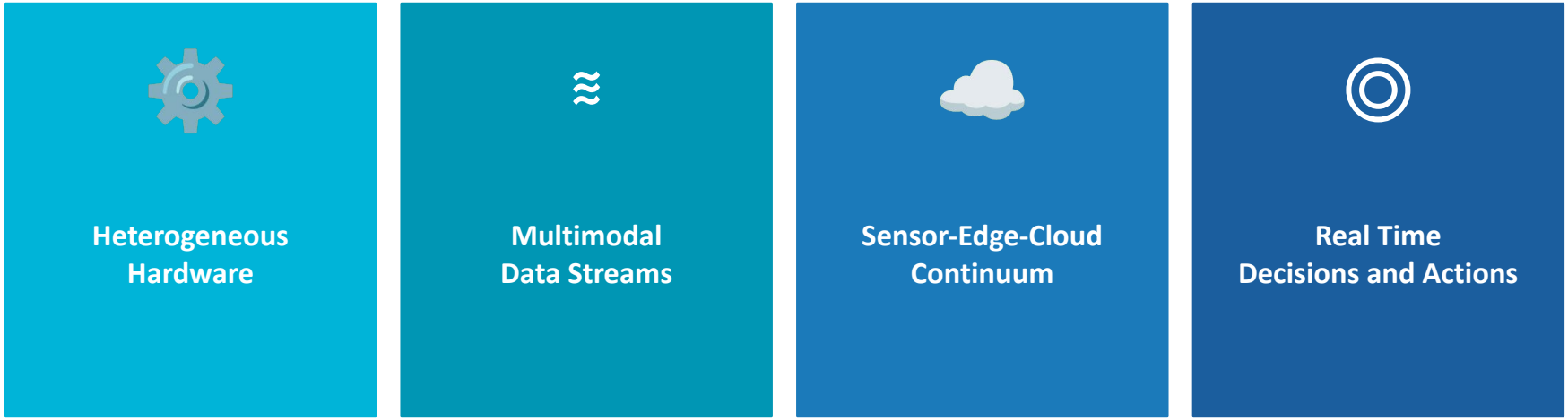
Flink Forward

Agenda

1. Stream Data Management
2. Recap: Apache Flink
3. **NebulaStream – Vision & History**
4. Current Applications
5. Design & Architecture
6. Open Source Status & Roadmap
7. Research Topics
8. Summary & How to Get Involved

NebulaStream – Vision

Making stream processing easy, scalable, and powerful — everywhere from sensor to data center



The NebulaStream Platform for Data and Application Management, Zeuch et al., CIDR 2020
NebulaStream - Data Stream Processing in Massively Distributed, Heterogeneous, Volatile Environments, Markl, DEBS 2024

NebulaStream – Design Goals

Ease-of-Use

Out-of-the-box functionality for the analysis of multi-modal, multi-frequency data streams.

Special operators for alignment, signal processing, time-series analysis, complex event processing, and inference.

Extensibility

Enable users to enhance NebulaStream and build custom applications.

Easy integration of custom data connectors, formats, operators, and optimizations into the system.

Efficiency

Hardware-tailored code generation & adaptive distributed processing on heterogeneous devices.

Low latency, high-throughput, and low footprint for real-time applications.

The NebulaStream Platform for Data and Application Management, Zeuch et al., CIDR 2020
NebulaStream - Data Stream Processing in Massively Distributed, Heterogeneous, Volatile Environments, Markl, DEBS 2024

Query Compilation for Streaming Queries

Philipp M. Grulich

Query Compilation for Stream Processing

- Adaptive query compilation for stream processing
- Task-based execution

Efficient Stream Processing Through Adaptive Query Compilation, Grulich et al., SIGMOD 2020

Query Compilation for Polyglot Queries

- Unified representation for polyglot queries
- Holistic optimizations

Efficient Execution of Polyglot Queries, Grulich et al., VLDB 2021

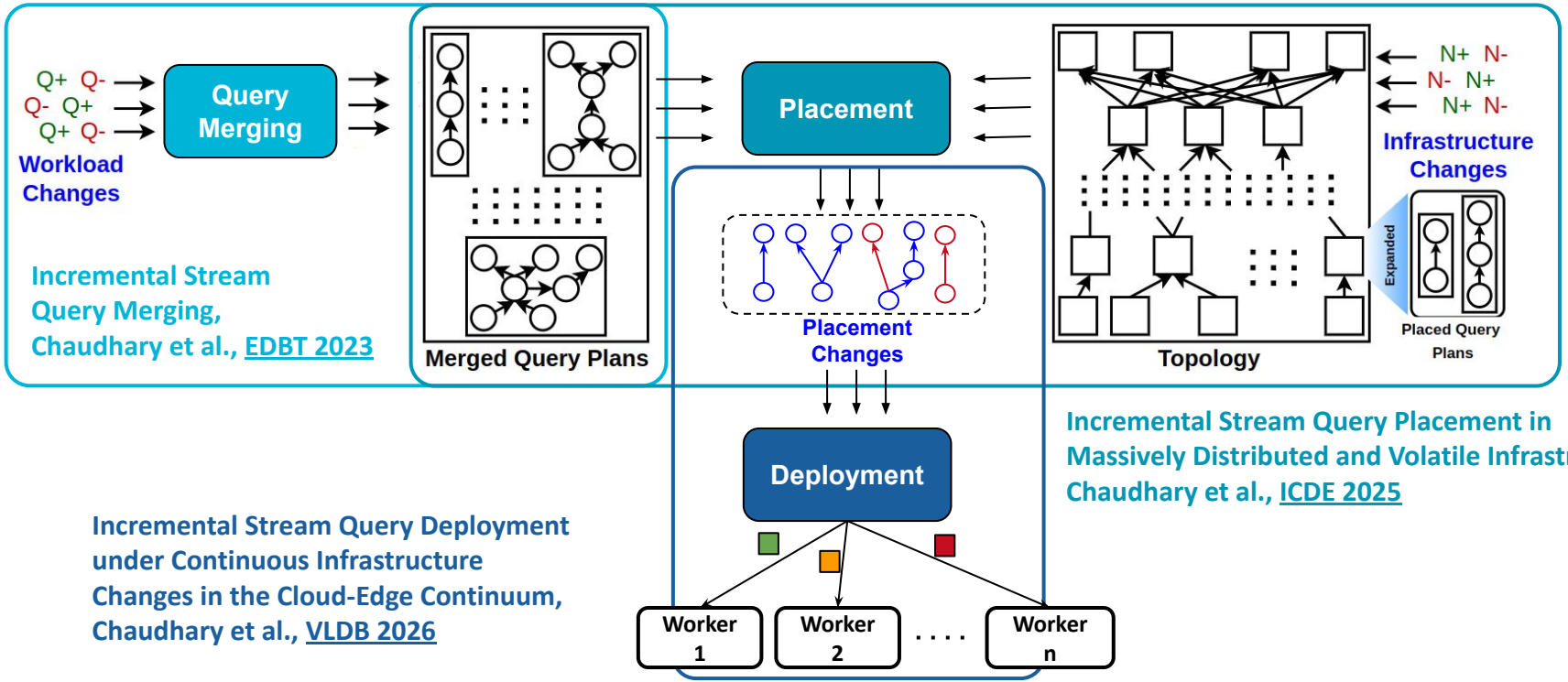
Query Compilation without Regrets

- Interpretation-based operator interface
- Trace-based, multi-backend JIT compiler

Query Compilation Without Regrets, Grulich et al., SIGMOD 2024

Optimizing Streaming Queries

Ankit Chaudhary



Incremental Stream Query Merging, Chaudhary et al., EDBT 2023

Incremental Stream Query Placement in Massively Distributed and Volatile Infrastructures, Chaudhary et al., ICDE 2025

Incremental Stream Query Deployment under Continuous Infrastructure Changes in the Cloud-Edge Continuum, Chaudhary et al., VLDB 2026

Fault-Tolerance

Anastasiia Kozar

Fault Tolerance Placement

- ➔ Treat fault tolerance as an operator
- ➔ Include fault tolerance in the cost estimation
- ➔ Fault tolerance integrated with the query optimizer

Fault Tolerance Placement in the Internet of Things,
Kozar et al., SIGMOD 2024

Hybrid Network-Aware Failure Recovery

- ➔ Redundant execution for zero-downtime recovery
- ➔ Efficient duplicate detection for correctness
- ➔ Dynamic load balancing for volatile environments

Meerkat: Scalable, Network-Aware Failure Recovery for the Internet of Things,
Kozar et al., VLDB 2026

Stream Embedded Checkpointing

- ➔ In-band checkpoint transfer over the data plane
- ➔ Chunk-level deduplication
- ➔ Lightweight rollback recovery in disaggregated environments

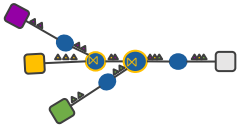
Paper Under Submission,
Kozar et al.

Complex Event Processing

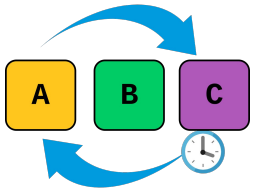
Ariane Ziehn

Visibility

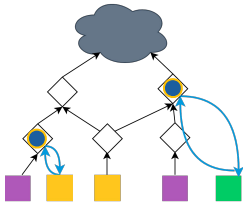
PATTERN A SEQ B SEQ C
WITHIN SLIDING [a.ts, size 20 MIN, slide 1 MIN]
INTO <sinks>

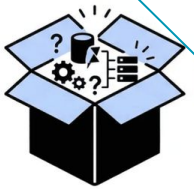


Logical Freedom



Physical Execution





1 Transform → 2 Reorder → 3 Deploy

Enabling Complex Event Processing in NebulaStream, Ziehn et al., EDBT 2025

Bridging the Gap: Complex Event Processing on Stream Processing Engines, Ziehn et al., EDBT 2024

Unraveling the Impact of Window Semantics: Optimizing Join Order for Efficient Stream Processing, Ziehn et al., VLDB 2025

Joint Optimization of Operator Placement and Communication for Distributed Stream Processing, Ziehn et al., DEBS 2026

Agenda

1. Stream Data Management
2. Recap: Apache Flink
3. NebulaStream – Vision & History
4. **Current Applications**
5. Design & Architecture
6. Open Source Status & Roadmap
7. Research Topics
8. Summary & How to Get Involved

Earlier Proof of Concept Projects

NebulaStream

- **Predictive Maintenance**
  paper mills · industrial sensor data streams
- **Smart Water Metering**
  distributed IoT infrastructure
- **Audiovisual Surveillance**
  license plate analysis · gunshot detection
- **Medical Wearables**
  privacy-aware ECG vest processing
- **Motorsport Bike Tuning**
  mobile edge · riding skill classification

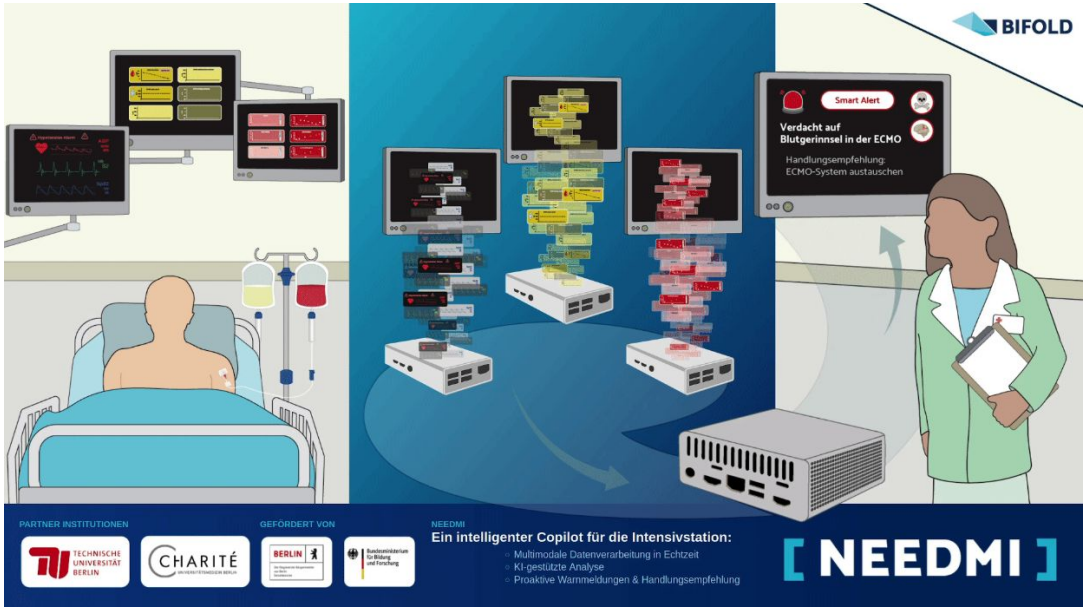
Current Pilots

NebulaStream

- **Smart ICU**
  intelligent co-pilot for intensive care · multi-modal real-time analytics and smart alerts
- **Smart Transportation**
  predictive maintenance & emergency detection · time-series real-time analytics and smart alerts
- **Smart Manufacturing**
  process quality control · multi-modal real-time analytics with ML inference and smart alerts

Smart Intensive Care Unit (ICU)

NebulaStream



NebulaStream: An Extensible, High-Performance Streaming Engine for Multi-Modal Edge Applications,
Michalke et al., SIGMOD 2025 (Demo)

Goal

- Intelligent Co-Pilot for ICU staff that sends proactive alerts when detecting deteriorating patient conditions

Problem

- Resource-constrained environment
- Multimodal vital and technical signals (e.g., blood pressure, ECG, breathing patterns, body temperature, signals from medical devices, e.g., ECMO unit)
- Real-time event detection

Solution with NebulaStream:

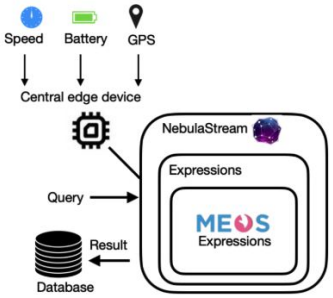
- Push down processing towards sensors to reduce network traffic
- ML inference close to the source

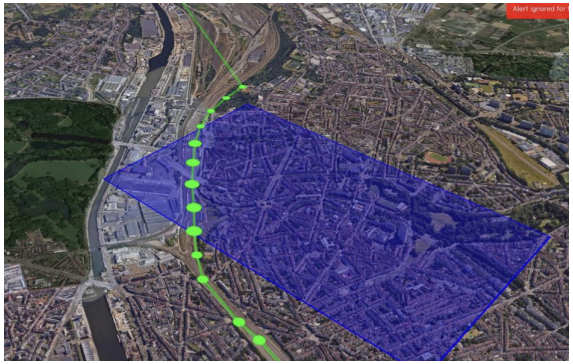
Status:

- Deployed live for a clinical study

Smart Transportation

NebulaStream





Mobility Stream Processing on NebulaStream and MEOS,
Duarte et al., SIGMOD 2025 (Demo)
Mobility Trajectory Data Stream Processing Beyond the Cloud,
Duarte et al., EDBT 2026

Goal:

- Timely alerting (e.g., predictive maintenance, faster emergency response)

Problem:

- Real-time analytics onboard of trains
- Low-end edge devices on moving trains with flaky network connection
  make timely alerting challenging

Solution with NebulaStream:

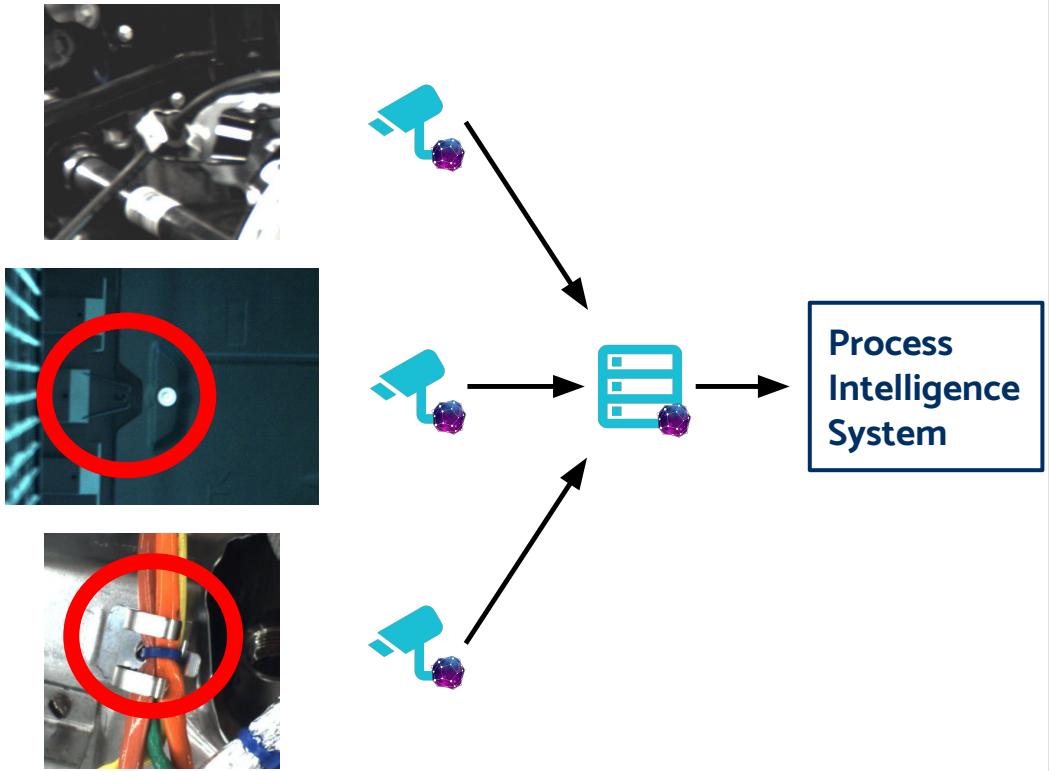
- Push down processing towards sensors to reduce time-to-reaction
- Queries with time-series and spatiotemporal operators

Status:

- Deployment and validation at SNCB headquarters

Smart Manufactoring

NebulaStream



Goal:

- Near real-time anomaly detection in car manufacturing based on video streams

Problem:

- Transmission of large data (e.g., videos) is prohibitive
- Detection-to-reaction latency in the cloud is too high

Solution with NebulaStream:

- Push ML inference closer to the source to reduce data transfer
- ML model deployment and update managed by NebulaStream

Status:

- PoC demo with a business process mining platform provider

Desiderata for Further Use Cases

NebulaStream

High Velocity Streams

millions of events/second, even on small edge devices

Multi-Modal Processing

efficient parsing, ingestion, and processing of arbitrary data formats

Diverse Workloads

stream processing, relational operators, plugin infrastructure for sources, sinks, and functions

Large Environments

1000s of nodes, compute push-down to any device

Heterogeneous Environments

low-end to high-end, any architecture

ML Inference

integration of ML models into streaming queries, native inference, easy ML model deployment & update

Volatile Environments

nodes appear, move, and disappear from the topology

High Privacy Demands

privacy-preserving processing enforced on-device

Dynamic Environments

dynamic adjustment to changing workloads

The NebulaStream Platform for Data and Application Management, Zeuch et al., CIDR 2020
NebulaStream - Data Stream Processing in Massively Distributed, Heterogeneous, Volatile Environments, Markl, DEBS 2024

Agenda

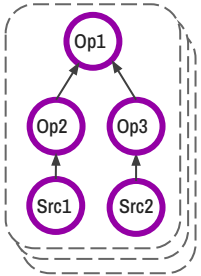
1. Stream Data Management
2. Recap: Apache Flink
3. NebulaStream – Vision & History
4. Current Applications
5. **Design & Architecture**
6. Open Source Status & Roadmap
7. Research Topics
8. Summary & How to Get Involved

Architecture – Distributed (Sensor-Edge-Cloud Continuum)

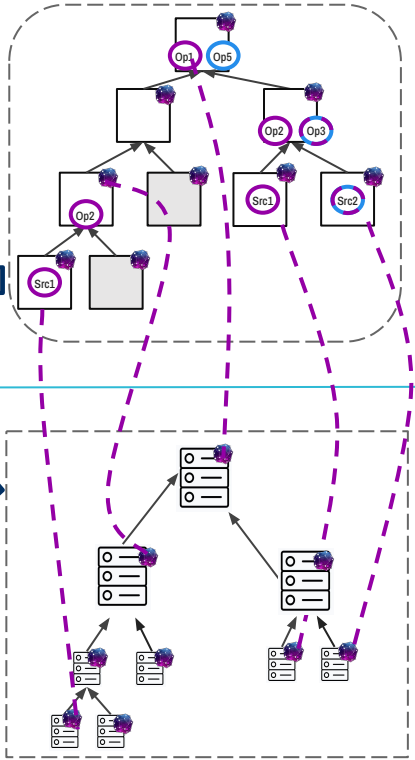
Declarative Query Language

```
SELECT SUM(value)
FROM INFER_MODEL(deviceIden, smart_meter)
WINDOW TUMBLING(timestamp, size 1 day)
GROUP BY customerId, deviceType;
```

Global Query Optimizer



Execution Network

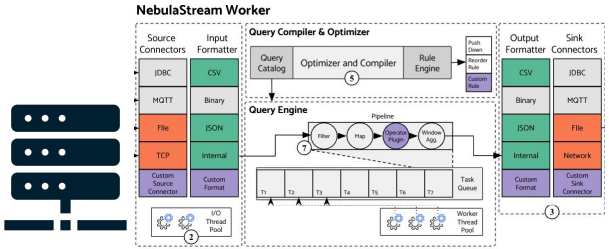


Hardware-tailored Code Generation

Compilation-time

Runtime

Efficient Query Execution Engine

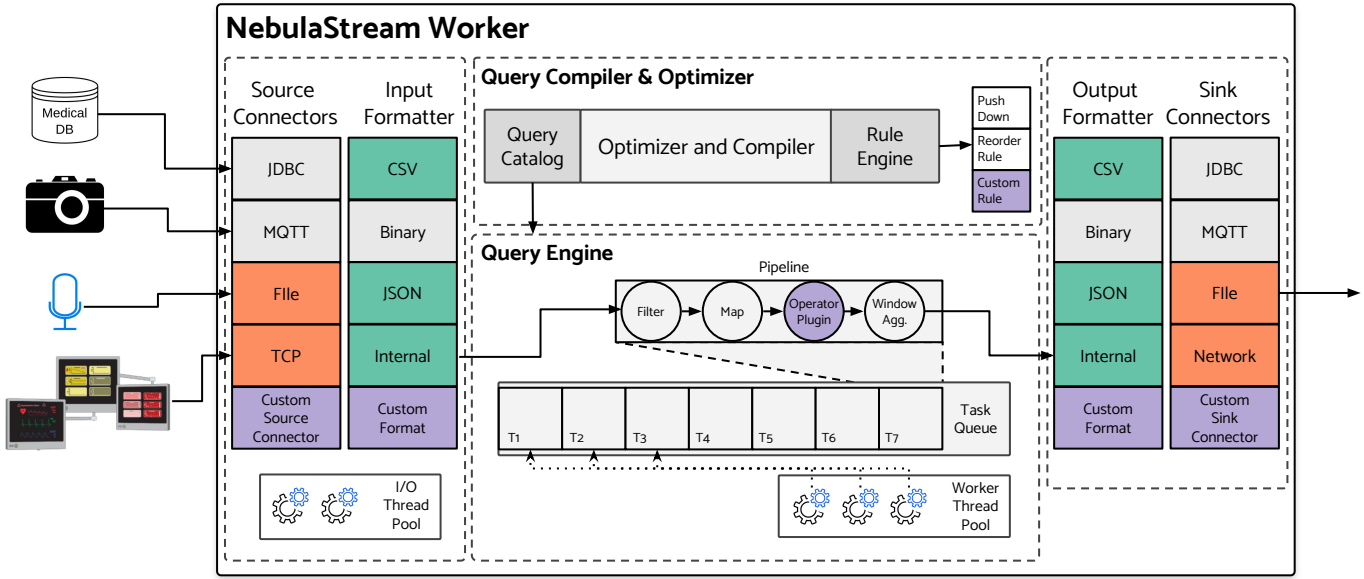


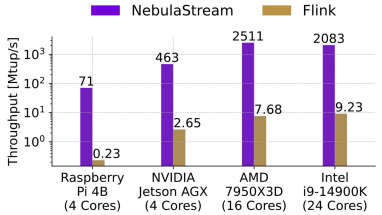
Worker(s)

Geo-distributed In-network Processing

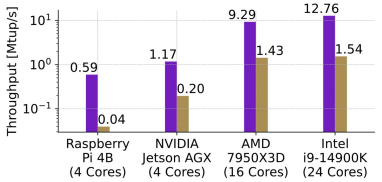
NebulaStream: An Adaptive and Efficient Multi-query Stream Processing Engine, Schubert et al., ICDE 2026

Single Worker Architecture





(b) Filter Query (*NM2*).



(d) Single Join (*NM8*).

NebulaStream: An Adaptive and Efficient Multi-query Stream Processing Engine, Schubert et al., ICDE 2026

NebulaStream – Queries

Sources

File (CSV/JSON) · TCP · Inline · Generator

Sinks

File (CSV/JSON) · Inline · Network · Print

Operators

Selection · Projection · Map · Union · Join · Inference

Windows

(keyed) Tumbling/Sliding ·
SUM · COUNT · AVG · MIN · MAX · MEDIAN

Functions

Arithmetic · Boolean · Comparison
String Concat · Unix Timestamp

Data Types

Basic Types · Nullable
Variable-sized Data / Strings

SQL-like

selection + projection + map

```
SELECT id, value + UINT64(10) AS bumped, timestamp
FROM stream
WHERE value > UINT64(5)
INTO Print();
```

keyed tumbling window aggregation

```
SELECT start, end, id, SUM(value) AS total
FROM stream
GROUP BY id
WINDOW TUMBLING(timestamp, size 1 sec)
INTO out;
```

inner join on multiple keys in a window

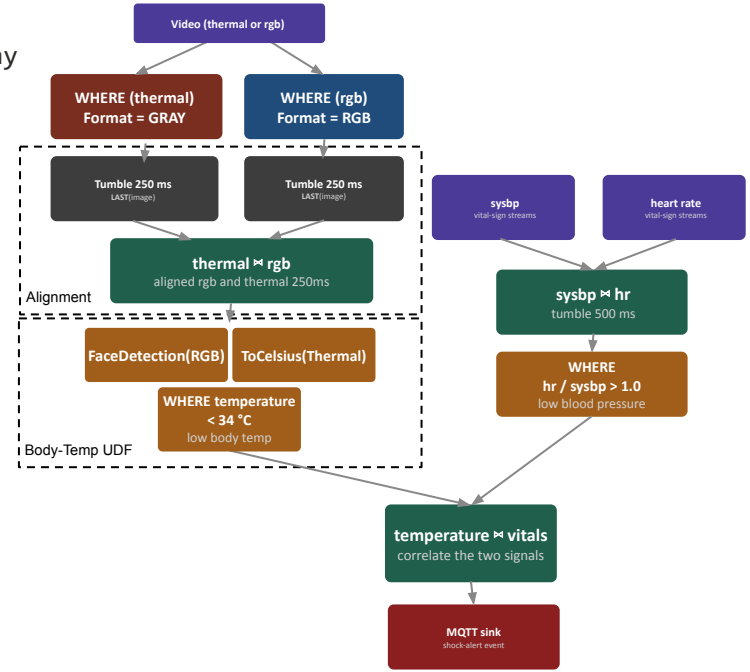
```
SELECT *
FROM stream AS s1
INNER JOIN stream2 AS s2
  ON s1.id = s2.id2 AND s1.value = s2.value2
WINDOW TUMBLING(timestamp, size 1 sec)
INTO File("out.csv" AS `SINK`.`FILE_PATH`, "CSV" as
`SINK`.`OUTPUT_FORMAT`);
```

Example

Input Data (Thermal Camera, Patient Monitor)

```
SELECT timestamp, VARSIZED("SHOCK") AS alertType
FROM (
  SELECT ts_gray AS timestamp
  FROM (
    SELECT LAST(img) AS img_gray, LAST(ts) AS ts_gray
    FROM video WHERE pixelFormat == VARSIZED("GRAY")
    WINDOW TUMBLING(timestamp, size 250 ms)
  )
  INNER JOIN (
    SELECT LAST(img) AS img_rgb, LAST(ts) AS ts_rgb
    FROM video WHERE pixelFormat == VARSIZED("RGB")
    WINDOW TUMBLING(timestamp, size 250 ms)
  ) ON ts_gray = ts_rgb
  WINDOW TUMBLING(ts_gray, size 250 ms)
  WHERE Celsius(MAX(ROI(img_gray,
    FaceDetection(img_rgb)))) < FLOAT64(34.0)
)
INNER JOIN (
  SELECT sysbp_ts AS bp_ts
  FROM sysbp
  INNER JOIN hr ON TRUE
  WINDOW TUMBLING(sysbp_ts, size 500 ms)
  WHERE hr_value / sysbp_value > FLOAT64(1.0)
) ON timestamp = bp_ts
WINDOW TUMBLING(timestamp, size 1000 ms)
INTO mqtt;
```

Query*



Query Plan (simplified)

*Discussion of the full Smart ICU Setting: Michalke et al., SIGMOD (Demo) 2025

Agenda

1. Stream Data Management
2. Recap: Apache Flink
3. NebulaStream – Vision & History
4. Current Applications
5. Design & Architecture
6. **Open Source Status & Roadmap**
7. Research Topics
8. Summary & How to Get Involved

NebulaStream Development

- NebulaStream GitHub organization
- Multiple repositories
- ~241 organisation members

NebulaStream

NebulaStream is a general purpose, end-to-end data management system for the IoT.

45 followers Berlin http://nebula.stream @nebulastream

https://github.com/nebulastream/nebulastream

Repository Statistics

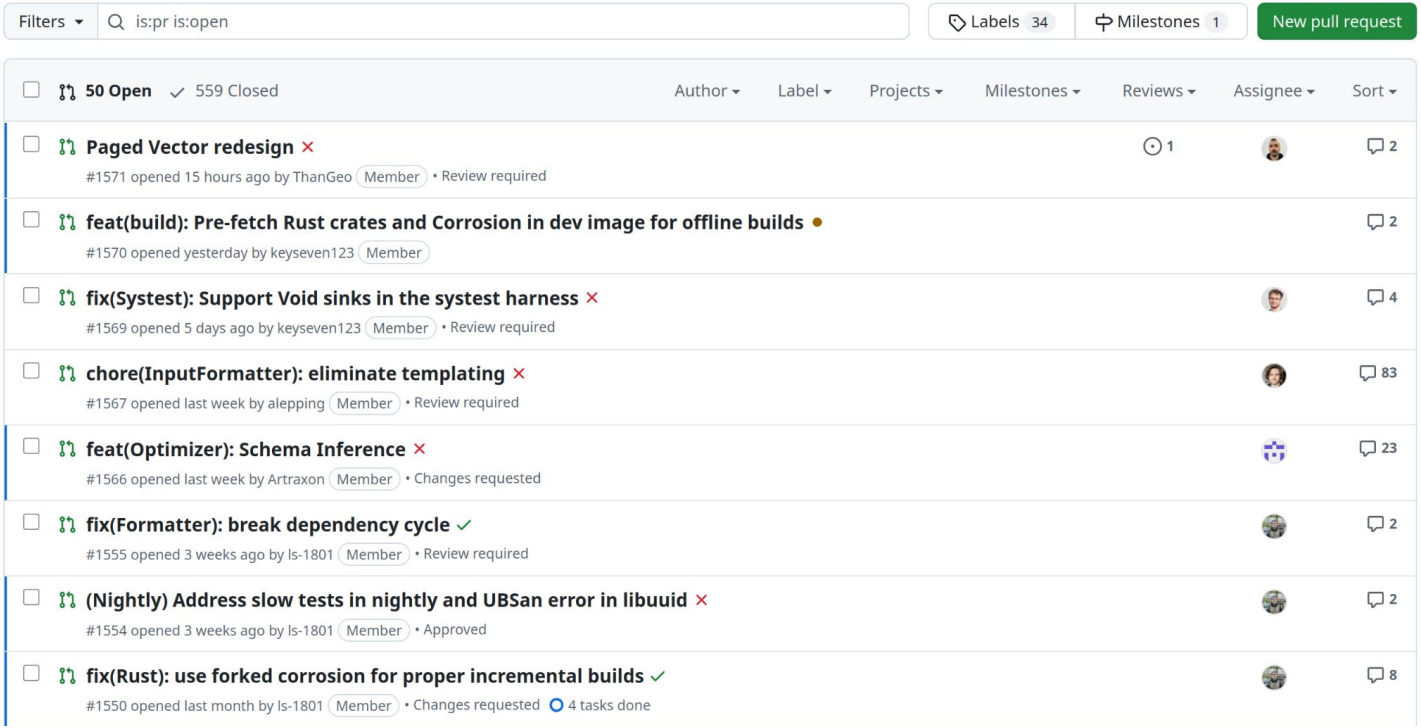
NebulaStream

- > 200K lines of code
- ~ 20K total commits
- 90 GitHub Stars
- 22 active contributors (last month)

https://github.com/nebulastream/nebulastream

Development Process (Pull Request)

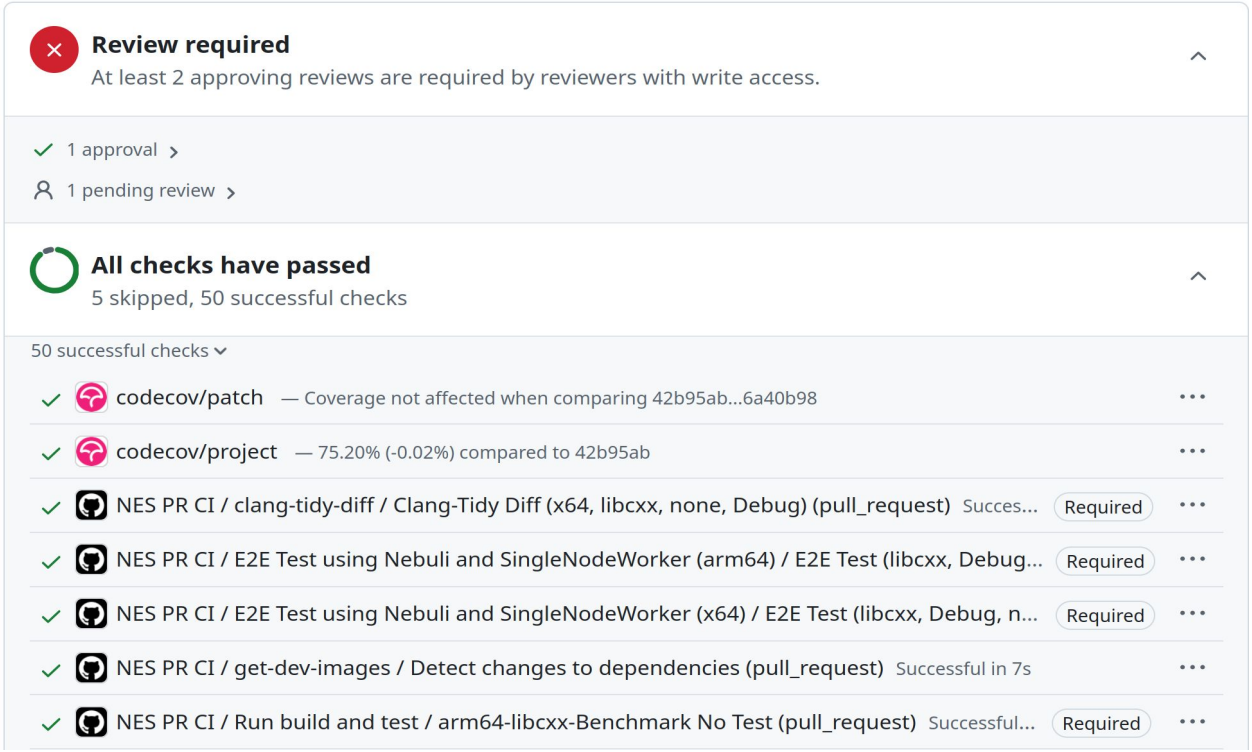
NebulaStream



NebulaStream uses GitHub's continuous integration functionality.

Example Pull Request

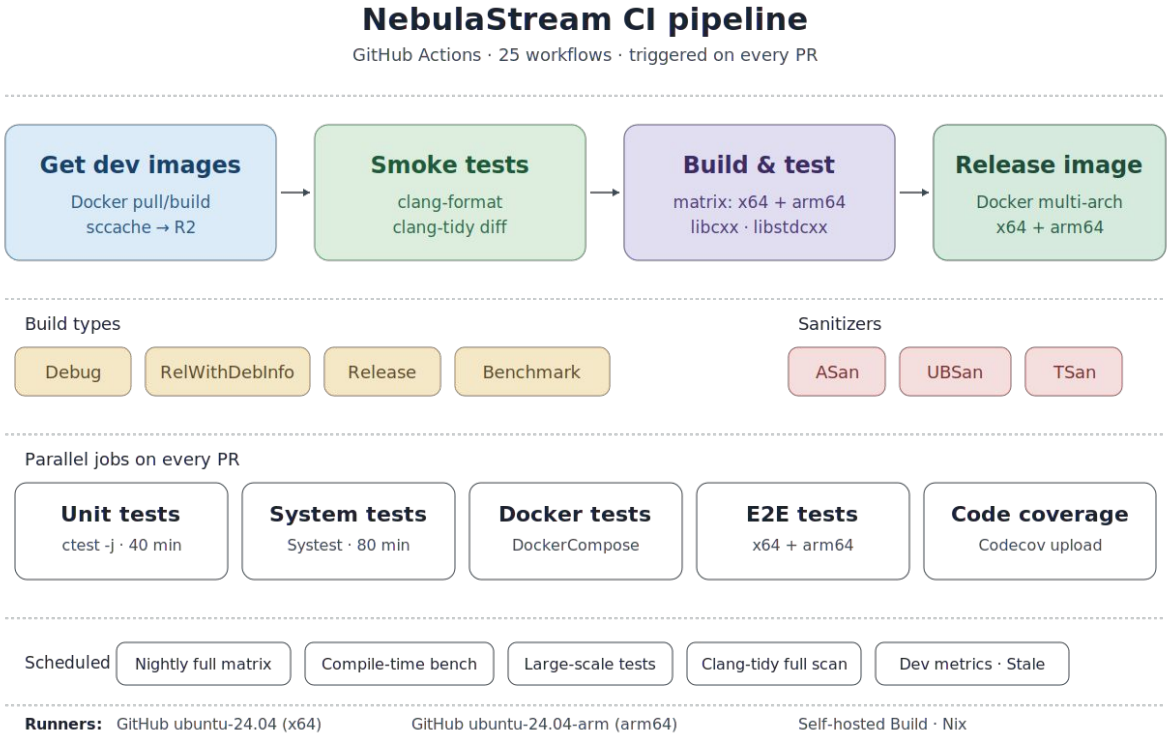
NebulaStream



Every pull request is checked on different architectures & operating systems.

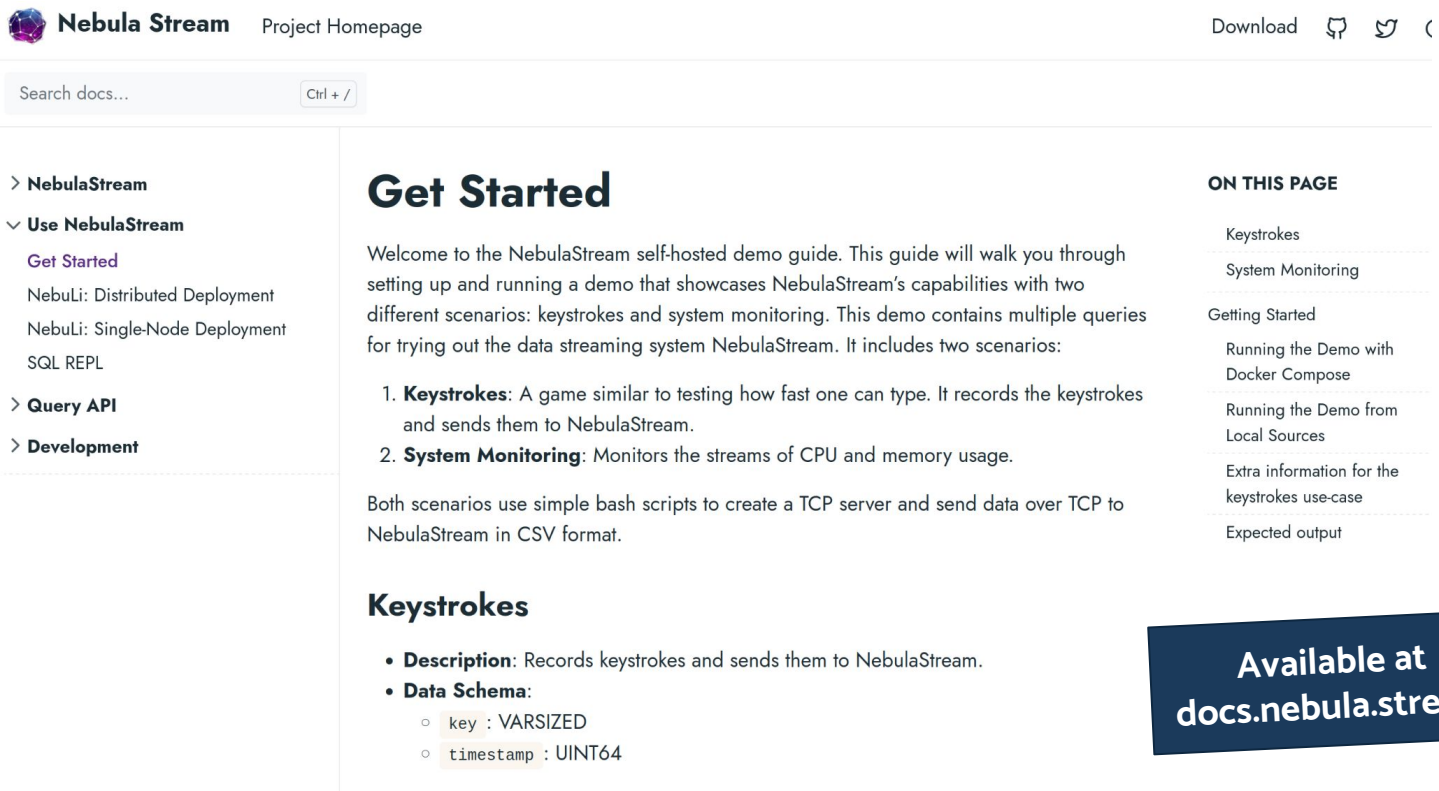
Release Process

NebulaStream



https://github.com/nebulastream/nebulastream

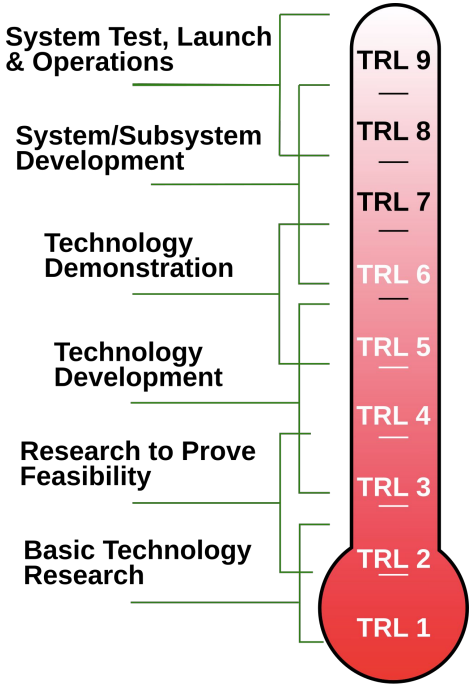
Documentation & Tutorial



Available at docs.nebula.stream

NebulaStream has an extensive online documentation.

Technology Readiness Level



TRL 7: Prototype is tested in an operational environment

TLR 6: Technology has been demonstrated in relevant environment

NebulaStream is between TRL 6 and TRL 7.

Roadmap - Upcoming Features

NebulaStream

Dynamic Adaptive Optimization
Re-plan queries as workloads drift
QUERY OPTIMIZER

Distributed Statistics Collection
System-wide statistics for the optimizer
QUERY OPTIMIZER

Checkpointing, Recovery & Replay
Fault-tolerance & time-travel for queries
RUNTIME

Decentralized Control Plane
Scale orchestration to millions of devices
CONTROL PLANE

Streaming Algebra
Formal query semantics & rewrites
LANGUAGE / API

Distributed Benchmarking Framework
Reproducible testing and evaluation
TOOLING

Extensible Data Types
User-defined types as first-class citizens
LANGUAGE / API

Semantic Operators
LLM-backed operators in streaming queries
OPERATORS / ML

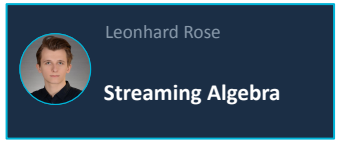
Stream Alignment
Synchronizing multi-frequency streams
OPERATORS / ML

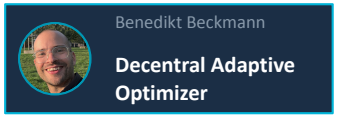
Agenda

1. Stream Data Management
2. Recap: Apache Flink
3. NebulaStream – Vision & History
4. Current Applications
5. Design & Architecture
6. Open Source Status & Roadmap
7. **Research Topics**
8. Summary & How to Get Involved

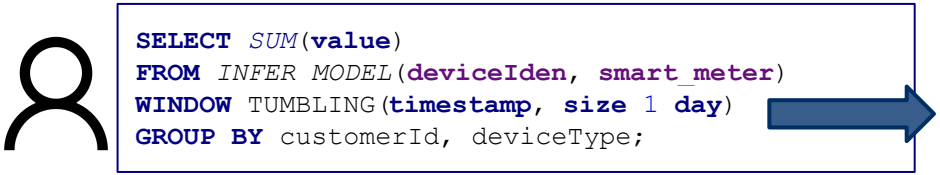
Research Topics

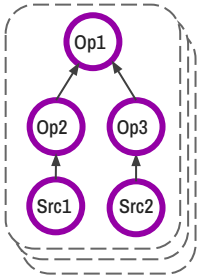
NebulaStream

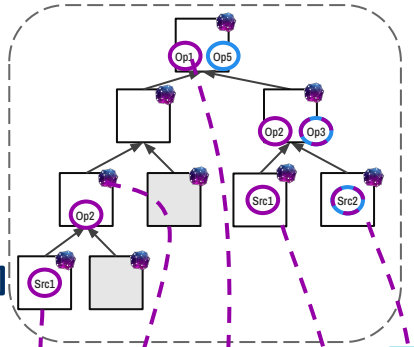




Execution Network

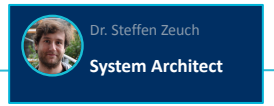






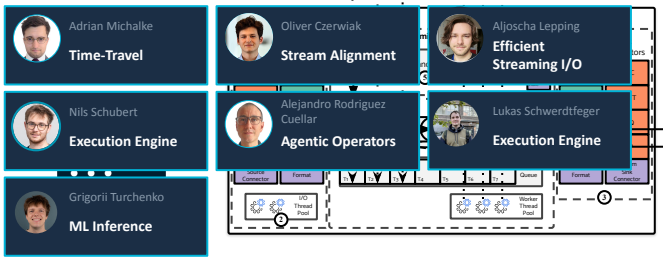
Compilation-time

Runtime



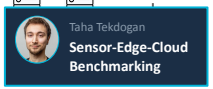
Hardware-tailored Code Generation

Efficient Query Execution Engine



Worker

Geo-distributed In-network Processing



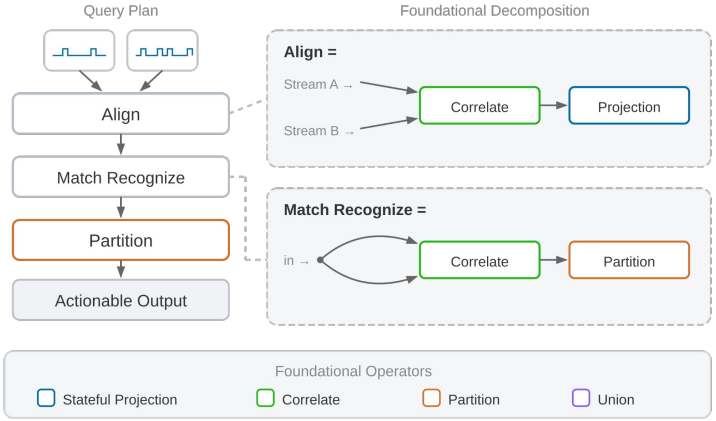
Programming for Streams

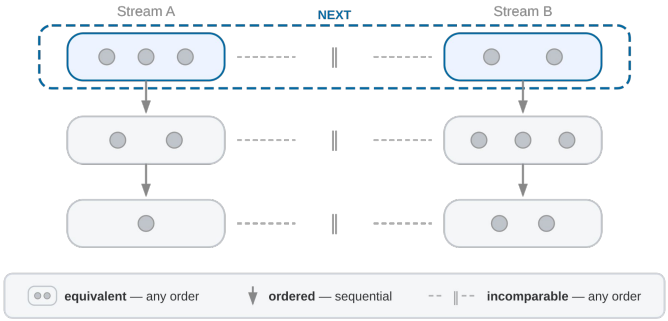
Leonhard Rose

Goal:
Enable the specification & automatic optimization of complex streaming applications for a massively distributed runtime

Underlying Research Problem:
Data analysis programs need to integrate and discretize data streams of different frequencies and modalities, analyze them with relational algebra and user-defined functions (UDFs), correlate events based on temporal or other relations, and trigger actions

Solution:
Define an algebra that encompasses operators for alignment & discretization of multi-frequency streams, relational, time series, complex event processing, and UDFs; identify optimizing transformations for these operators



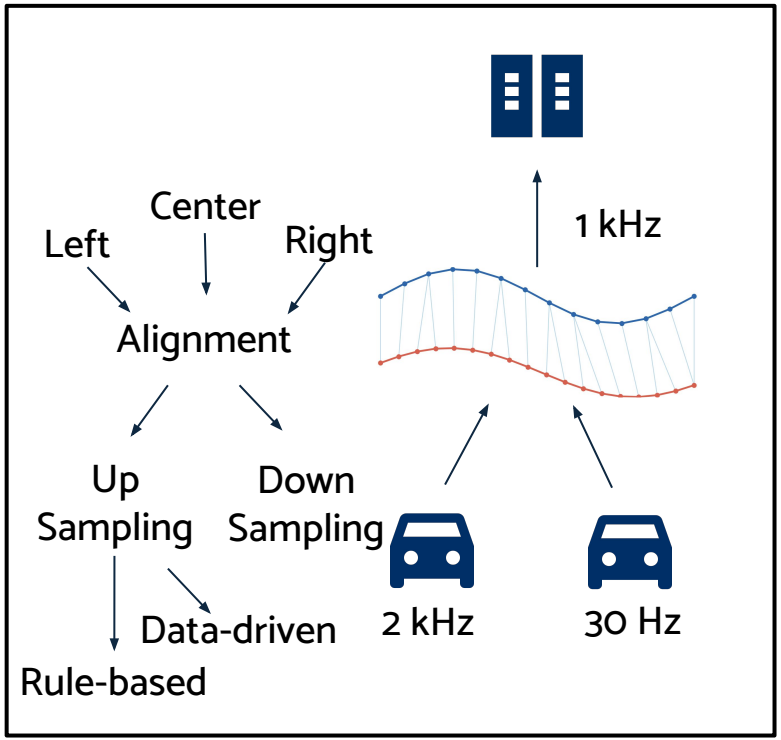


Stream Alignment

Goal:
A unified abstraction and efficient implementation for multi-modal, multi-frequency alignment of distributed data streams

Underlying Research Problem:
Data rates from devices are highly heterogeneous; current methods cannot (temporally) align data streams with diverse data rates automatically for subsequent joint processing

Solution:
An alignment operator that aggregates, interpolates, and extrapolates, to ensure proper (temporal) alignment of all streams involved in a query



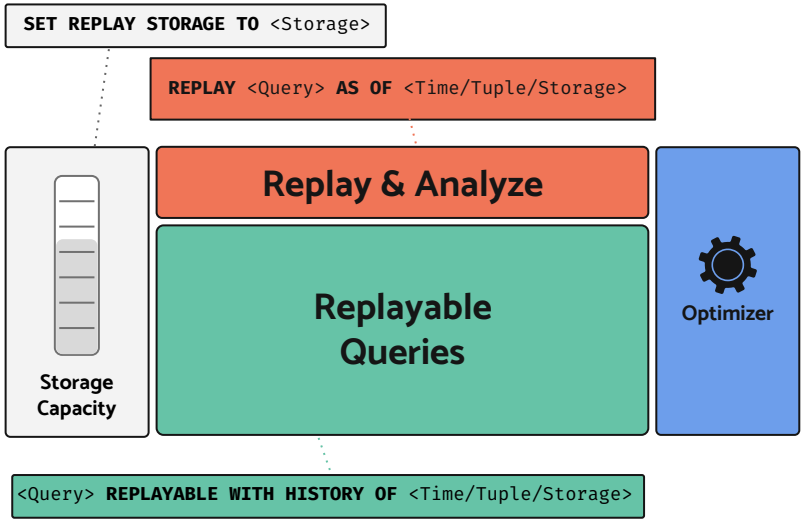
Declarative Time Travel

Adrian Michalke

Goal:
Declarative specification and efficient processing of time-travel for distributed data streams (e.g., for backtesting, simulation, fault-tolerance, debugging)

Underlying Research Problem:
How to efficiently replay data streams in massively distributed systems? How to enable replayability at minimal cost, i.e., predefined memory budget, latency, impact on the system, and storage capabilities?

Solution:
Add materialization operators to query plans based on declarative replay specifications; the optimizer decides placement



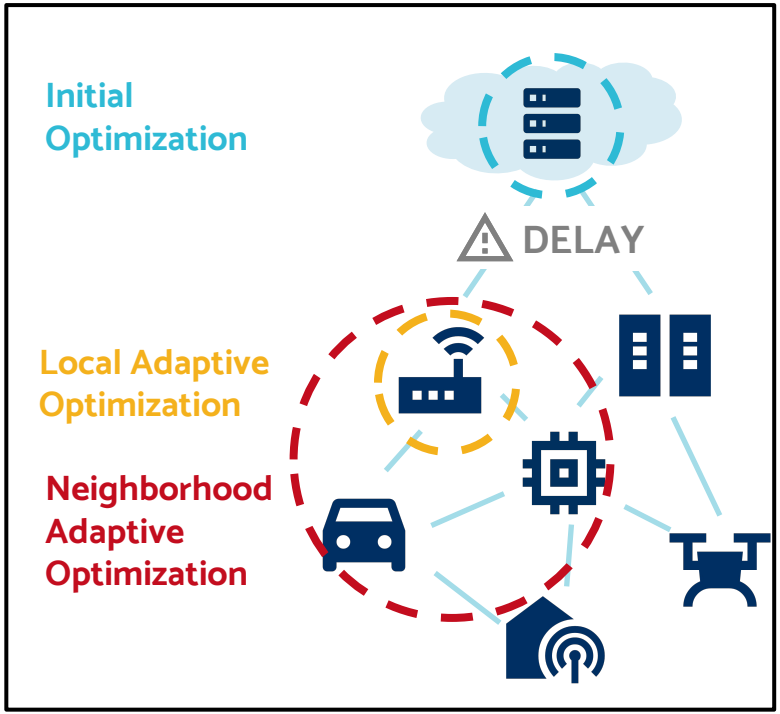
Distributed and Adaptive Query Optimization

Benedikt Beckermann

Goal:
Ensure continued and fast query execution in massively distributed edge deployments with changing data characteristics and volatile topologies

Underlying Research Problem:
How can we adapt the processing of streaming queries while avoiding latencies and bottlenecks introduced by centralized solutions?

Solution:
Decentrally (re-)optimize streaming (sub-)queries in neighborhoods



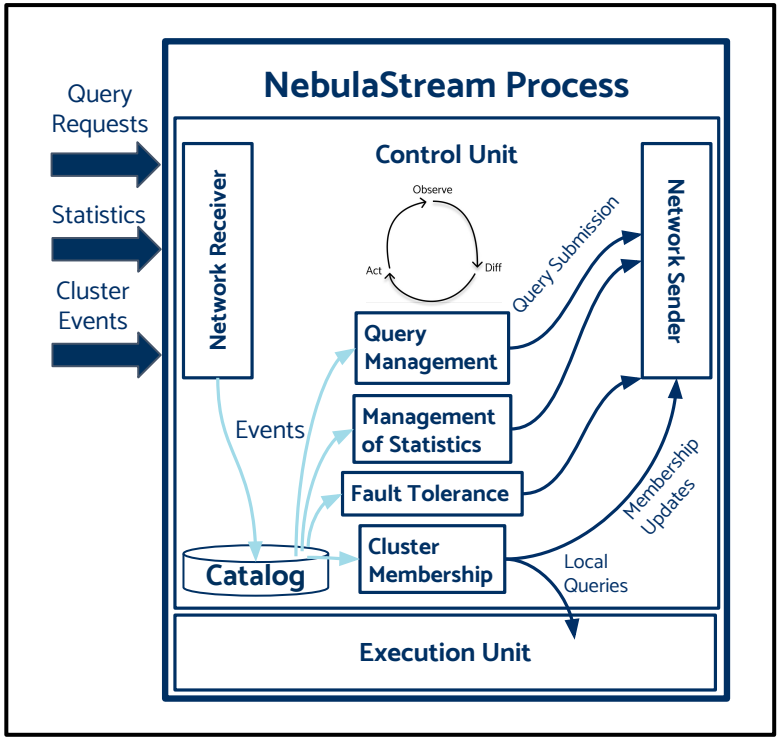
Decentralized Orchestration of Millions of Devices

Yannik Schröder

Goal:
Enable reliable, zero-downtime continuous stream processing in large, volatile, and heterogeneous topologies

Underlying Research Problem:
How can we detect failures in large-scale deployments fast and reliably?
How can we efficiently communicate membership changes to react quickly?
How can we recover from failures locally?

Solution:
Nodes observe their neighborhood and upon changes reconcile; new knowledge is gossiped to neighbors



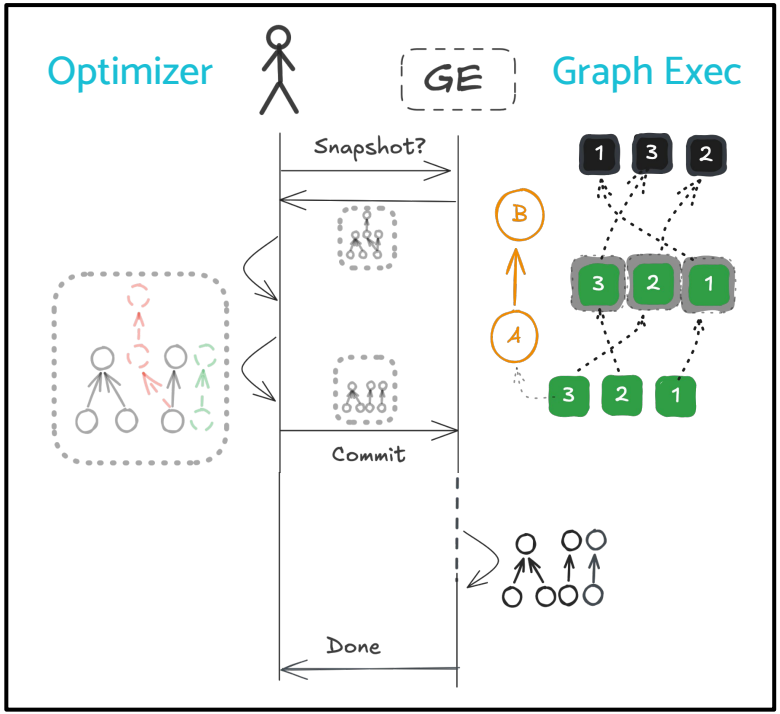
Adaptive Out-of-order Execution

Lukas Schwerdtfeger

Goal:
Provide multi-threaded out-of-order execution of thousands of stateful queries, to allow for runtime adaptation (e.g., rescaling, swapping plans)

Underlying Research Problem:
Reason about consistency of the execution graph under concurrent out-of-order execution, without blocking worker processes

Solution:
Snapshot-based (MVCC-like) interface for the optimizer, with transactional modifications to the execution graph



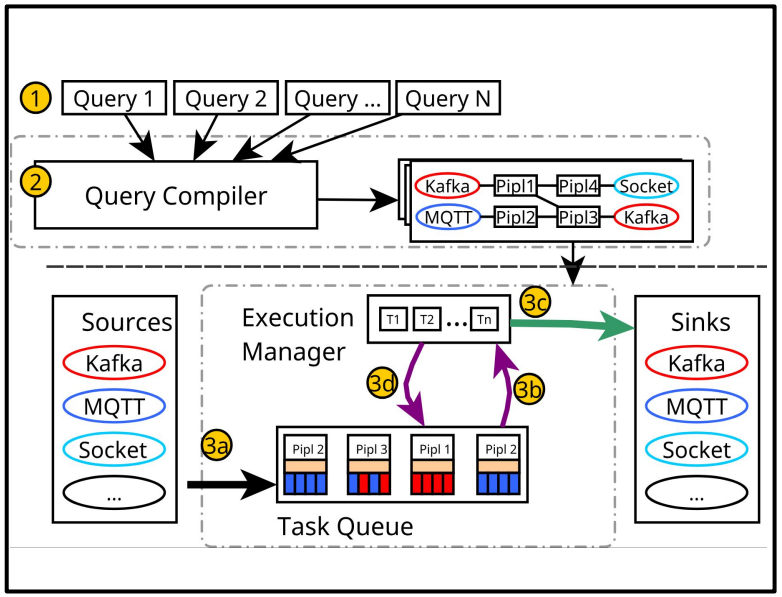
Multi-Query Execution


Nils Schubert

Goal:
Enable resource-efficient stream processing in a massively distributed and heterogeneous environment

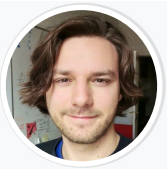
Underlying Research Problem:
How can hardware resources be dynamically allocated fairly in real-time for multiple concurrent queries under fluctuating ingestion rates?

Solution:
Adaptive, work-stealing, task-based execution with hardware-tailored code generation



NebulaStream: An Adaptive and Efficient Multi-query Stream Processing Engine, Schubert et al., ICDE 2026

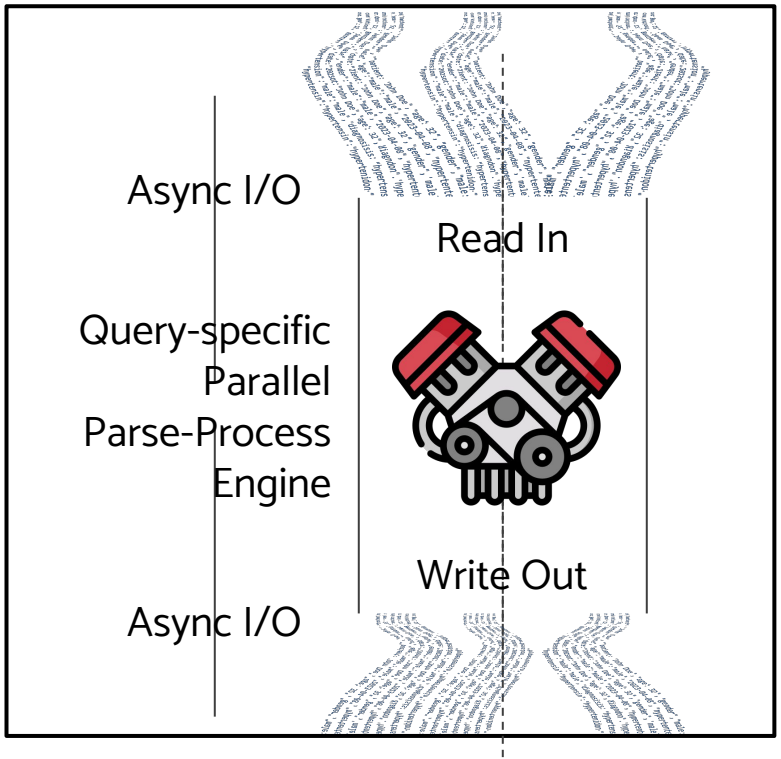
Efficient Streaming I/O



Goal:
An I/O framework for streaming workloads that keeps up with NebulaStream's compiled and highly efficient query engine

Underlying Research Problem:
Parsing external (e.g., text data) can quickly become *the* central bottleneck of a streaming query

Solution:
An ingestion framework that asynchronously ingests buffers, parses in parallel and fuses I/O-parsing deep into compiled pipelines



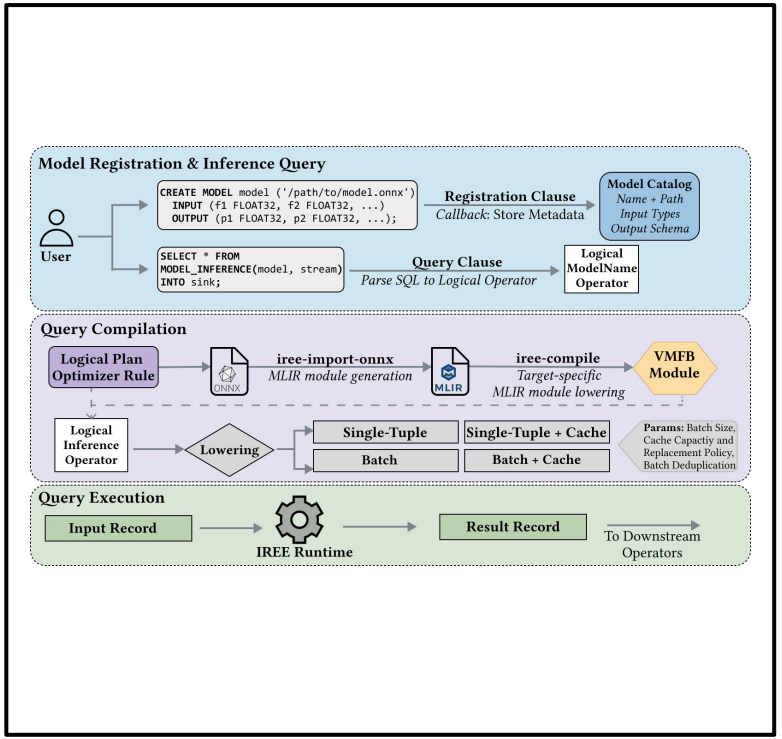
Efficient ML Inference for High-Velocity Data Streams

Grigorii Turchenko

Goal:
Enable resource-efficient, high-throughput ML inference to support deployment in volatile edge environments on heterogeneous hardware

Underlying Research Problem:
External model serving may not be possible due to latency or compliance constraints; how can we efficiently enable embedded ML inference beyond UDFs?

Solution:
Native integration of ML inference it into the system through several efficient physical inference operators (e.g., leveraging batching, prediction caching)



LLM-based Semantic Operators for Data Streams

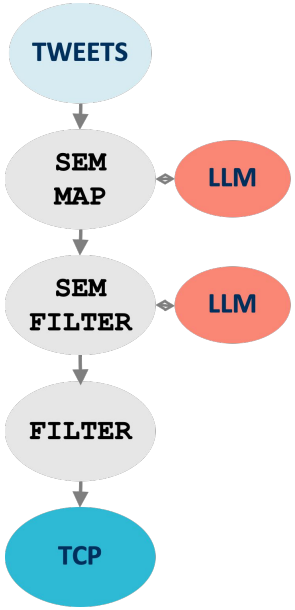
Alejandro Rodriguez Cuellar

Goal:
Provide semantic operators for high-velocity data streams to enable deep semantic reasoning

Underlying Research Problem:
LLM-based semantic operators in streaming systems create a trade-off between cost (expensive calls), latency (slow execution), memory (bounded state maintenance), and quality (probabilistic outputs)

Solution:
Interleave LLM-based semantic operators with relational relational logic to enable traditional optimizations (e.g., fusion, fission, reordering, and batching) and LLM-specific optimizations (e.g.,model selection, cascading, code synthesization)

```
SELECT
TWEET,
SEM_MAP('Extract
topic of the tweet',
TWEET) as topic
FROM TWEETS
WHERE
SEM_FILTER('Sentiment
of the tweet is
positive', TWEET)
AND TS>'2026-03-27'
INTO
TCP();
```



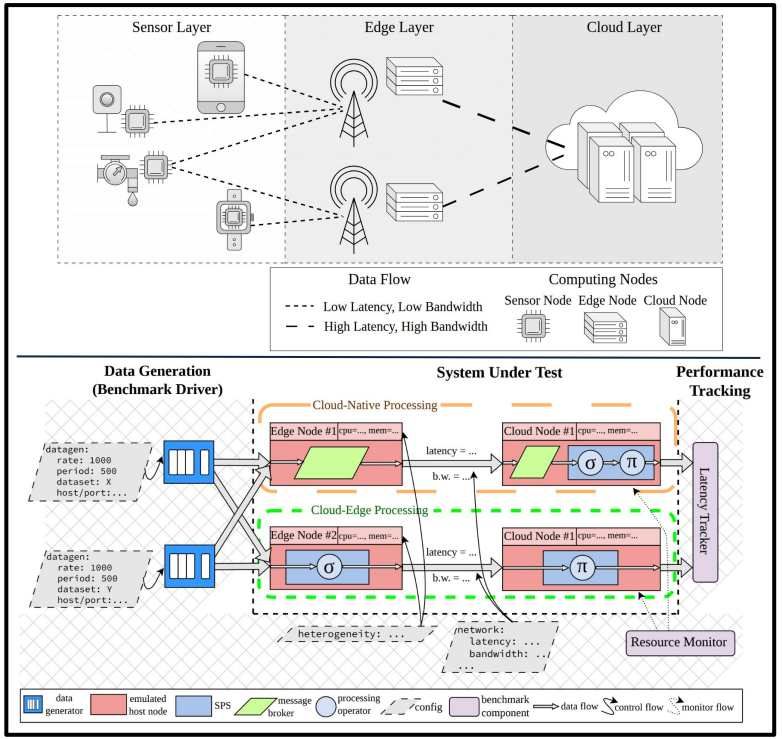
Benchmarking Stream Processing Systems in the Sensor-Edge-Cloud Continuum

Taha Tekdogan

Goal:
Enable systematic performance evaluation of stream processing systems in heterogeneous sensor-edge-cloud environments

Underlying Research Problem:
Existing streaming benchmarks focus on cloud-only processing; effects of distribution, network latency, and hardware heterogeneity on performance are neglected

Solution:
A benchmarking framework that models the characteristics of the sensor-edge-cloud continuum; tailored evaluation metrics and SUT definition



Open Research Topics

NebulaStream

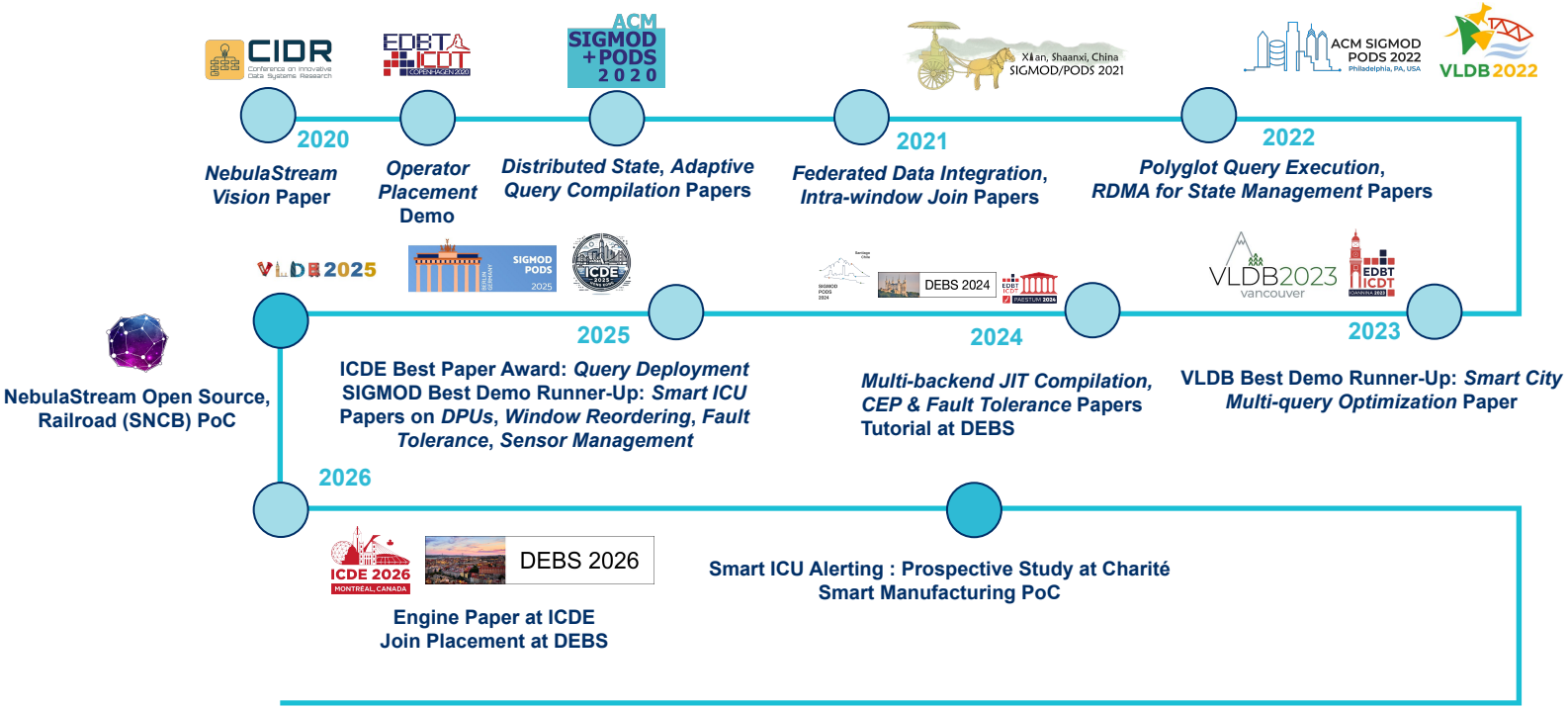
- **UDFs**
  polyglot user-defined functions
- **Microcontrollers**
  resource-aware code generation
- **Hardware Acceleration**
  GPU · FPGA · NIC offloading
- **Heterogeneous Storage**
  object store · TSDB · in-memory tiers
- **ML Training**
  continuous learning on streams
- **Feature Stores**
  stream features as first-class citizens
- **Vector Operators**
  similarity search on streams
- **Privacy**
  on-device processing · anonymization
- **Security**
  trusted execution · access control · encryption
- **Data Sovereignty**
  locality constraints as predicates
- **Debugging**
  explainability
- **Compliance-aware Planning**
  GDPR · HIPAA as optimizer constraints
- **… and many more**

Systems ML Trust

Agenda

1. Stream Data Management
2. Recap: Apache Flink
3. NebulaStream – Vision & History
4. Current Applications
5. Design & Architecture
6. Open Source Status & Roadmap
7. Research Topics
8. **Summary & How to Get Involved**

Innovation Pipeline
from Flink to NebulaStream



Basic Research

Applied Research

Commercialization

Team

TU Berlin / BIFOLD


Samira Akili


Benedikt Beckermann


Ankit Chaudhary


Alejandro Rodriguez Cuellar


Oliver Czerwiak


Thanasis Georgiadis


Philipp M. Grulich


Anastasiia Kozar


Aljoscha Lepping


Ricardo Martinez


Nikolai Merkel


Adrian Michalke


Leonhard Rose


Yannik Schröder


Nils Schubert


Lukas Schwerdtfeger


Taha Tekdogan


Grigorii Turchenko


Steffen Zeuch


Ariane Ziehn

Charité / BIFOLD


Christoph Falkensteiner


Kyle Krüger


Alexander Meyer


Tobias Röschl


Svea Wilkending

ULB / BIFOLD


Mariana Duarte

Want to learn more? Attend Nils' talk!

Nils Schubert

NebulaStream: An Adaptive and Efficient Multi-query Stream Processing Engine

Nils L. Schubert*, Lukas Schwerdtfeger*, Sara Schnaterbeck*,
Philipp M. Grulich*†§, Bonaventura Del Monte*†§, Steffen Zeuch*, Volker Markl*‡
BIFOLD / Technische Universität Berlin*, Snowflake Inc., Berlin, Germany†, DFKI GmbH‡
nils.schubert@dima.tu-berlin.de, {schwerdtfeger, s.schnaterbeck, steffen.zeuch, volker.markl}@tu-berlin.de
{philipp.grulich, ventura.delmonte}@snowflake.com

Poster: TKDE and Research Poster Session A
Tue 10:00–12:00
Location: 217 - Av. Viger

Talk: R14 - Stream Processing Engines and Architectures
Tue 15:30–17:00
Location: 208 - Rue Crescent

How to Get Involved

→ https://nebula.stream
→ github.com/nebulastream/nebulastream
→ Issues, pull requests, and discussions are welcome
→ Interest in new application scenarios: contact us!

Email: nebulastream@dima.tu-berlin.de

We are Hiring: Fully-funded PhD Student & Postdoc Positions

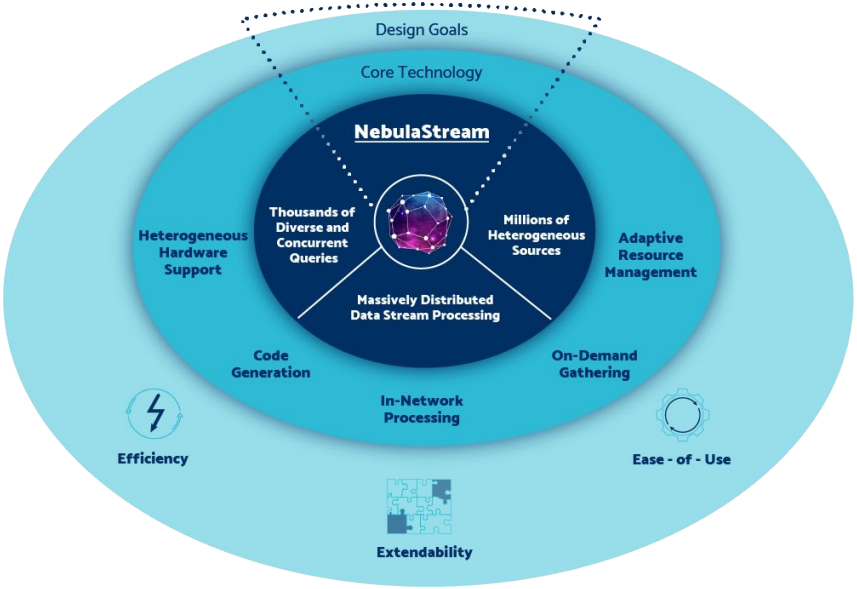
Reach out to Nils, Alexander, Steffen, Juan, or Volker during ICDE via Whova to meet and discuss
https://www.tu.berlin/en/dima/employment-opportunities
Send your application to jobs@dima.tu-berlin.de

NebulaStream Summary

RESEARCH

- **Overview**
  CIDR20 · JIoT20
- **Query Optimization**
  VLDB26 · ICDE25 · BTW25 · BTW23 · EDBT23 · JIoT21
- **Sensor Management**
  VLDB25 · ICDE25 · DEBS20
- **Complex Event Processing**
  EDBT25 · EDBT24
- **Query Compilation**
  SIGMOD24 · CIDR23 · VLDB21
- **Fault Tolerance**
  VLDB26 · SIGMOD24
- **Query Placement**
  VLDB24
- **Execution Engine**
  ICDE25
- **Use Cases**
  JIoT22 · SIGMOD21
- **Energy Efficiency**
  DaMoN21
- **Monitoring**
  JIoT21
- **Mobility**
  SIGSPATIAL25
- **Heterogeneous Hardware**
  VLDB25 · EDBT24

CORE TECHNOLOGY & DESIGN GOALS



PROJECT PARTNERS

SIEMENS · PIERER INNOVATION · spark works · UBITECH ubiquitous solutions · B · CHARITÉ UNIVERSITÄTSMEDIZIN BERLIN

WEBSITE nebula.stream

APPLICATIONS

- **Predictive Maintenance**
  Paper Mills
- **Smart Water Metering**
  Distributed Sensors
- **Audio/Video Surveillance**
  License Plates & Gunshots
- **Medical Wearables**
  Privacy-aware ECG Vests
- **Mobile Devices**
  Motorsport Tuning & Rider Skill
- **Smart Manufacturing**
  Industrial Automation
- **Smart ICU**
  Co-pilot for Intensive Care
- **Smart Transportation**
  Mobility Processing at the Edge
- **Assisted Surgeries**
  Non-invasive Heart Surgery Alignment