The NebulaStream Platform

Data and Application Management for the Internet of Things

BIFOLD

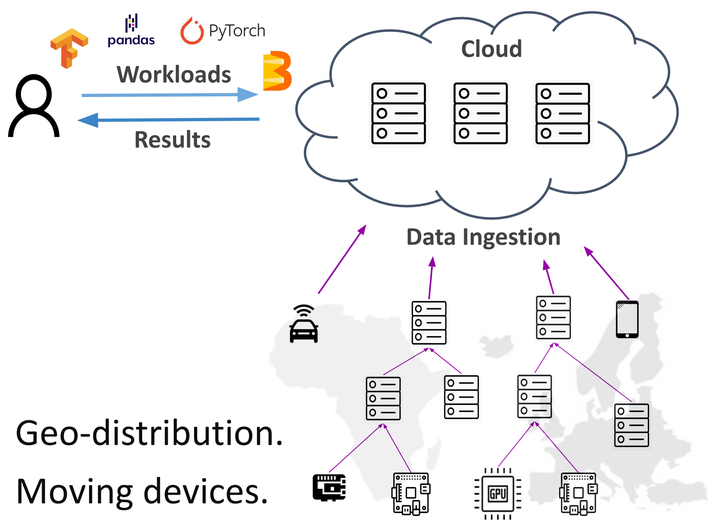
Overview

NebulaStream is a <u>scalable</u>, <u>adaptive</u>, and <u>efficient</u> data management platform for the Internet-of-Things

Research Goals:

- Efficient execution for thousands of <u>concurrently running queries</u>.
- Fast deployment to <u>massive</u> and <u>dynamic</u> topologies combining edge and cloud.
- Full utilization of <u>heterogeneous hardware</u> resources and accelerators.
- Support for complex analytical workloads involving stateful operators and UDFs.

Unifying Edge and Cloud



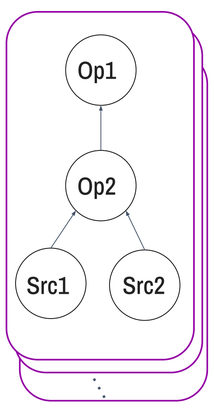
- Geo-distribution.
- Moving devices.
- Hierarchical topologies.
- Diverse use cases and workloads.
- Heterogeneous compute resources.
- Control over data acquisition.

An Example Application

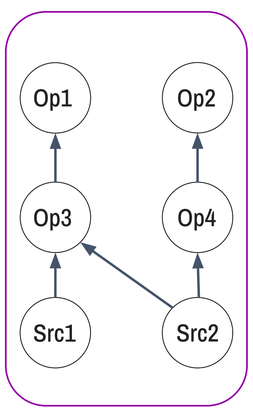

```
Query::from("smart_meter")
.filter(Attribute("type")=="electricity")
.window(TumblingWindow(days(1))
.byKey(Attribute("owner_id"))
.apply(Sum("value"))
```

Distributed Query Deployment Flow in NebulaStream

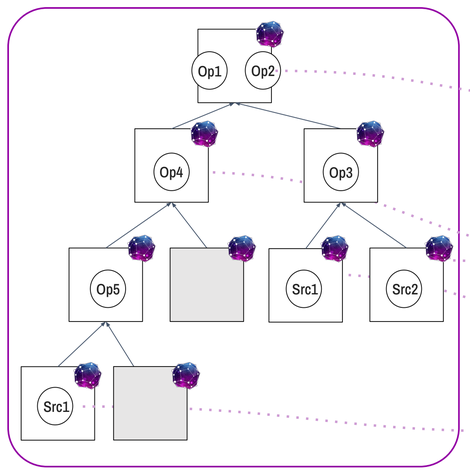
1 Single Query Optimization



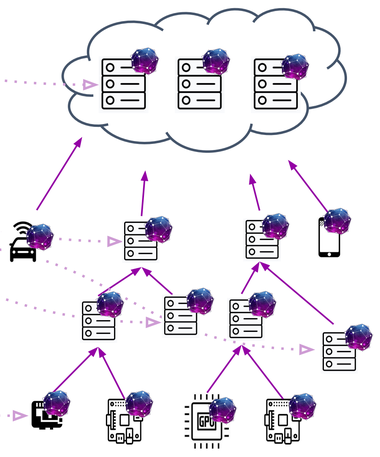
2 Query Merging



3 Query Placement



4 Query Execution



Visit our website

Learn about NebulaStream.
Try out our examples.
Join our project and collaborate with us!

NebulaStream
<u>https://nebula.stream</u>
nebulastream@dima.tu-berlin.de

How to optimize query plans?

Concurrently-running streaming queries often consume the same sources or perform similar tasks.

We use Semantic Stream Query Merging:

- To derive semantic operator signatures.
- To identify sharing opportunities via constraint solving even for syntactically different queries.

Query 1

```
Query::from("car")
.map(Attribute("speed") = Attribute("speed")*1.6)
.filter(Attribute("speed") > 100)
.map(Attribute("over_speed") = true)
.sink(...)
```

Query 2

```
Query::from("car")
.filter(Attribute("speed")*1.6 > 100)
.map(Attribute("speed") = Attribute("speed")*1.6)
.map(Attribute("over_speed") = true)
.sink(...)
```

Common Signature

Conditions: car.source == “car” and car.speed * 1.6 > 100
Columns: car.speed = car.speed * 1.6
car.id = car.id
car.loc = car.loc
car.over_speeding = true

Enable resource-efficient execution of semantically-equivalent queries

Where to place operators?

Unified edge/cloud environments consist of heterogeneous nodes with very different resources.

We explore different operator placement strategies:

<u>Constructive Approach:</u>

- Bottom Up
- Top Down

<u>Cost-based Approach:</u>

- Random Search
- Integer Linear Programming
- Genetic Algorithm

<u>Challenges:</u>

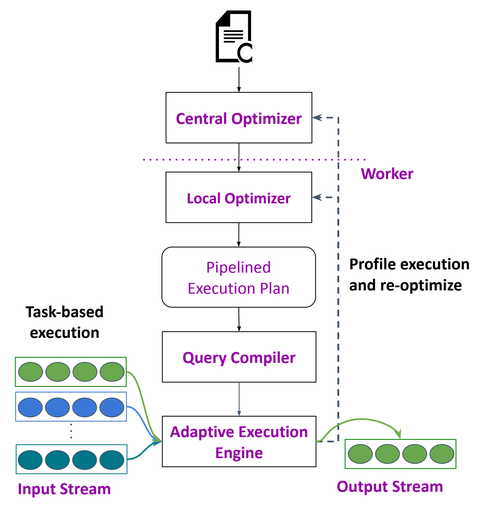
- **Millions of devices**
- **Resource heterogeneity**
- **Hierarchical infrastructure**
- **Geo-distributed data sources**
- **Changing data-characteristics and statistics**

Enable user to make trade-offs to speed-up query deployment

How to leverage heterogeneous resources?

State-of-the-art SPSs do not fully utilize available hardware resources.

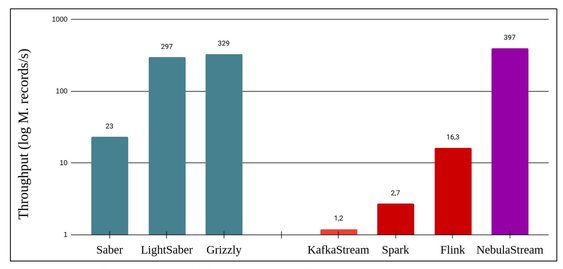
We rely on adaptive query compilation to specialize the execution to data and hardware characteristics.



Enable hardware-conscious and data-conscious query execution

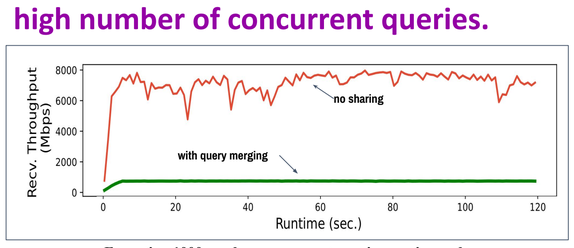
Evaluation

NebulaStream combines the performance of research prototypes with the generality of mature SPSs.



Executing the Yahoo Streaming Benchmark on eight cores.

Query merging is crucial to support a high number of concurrent queries.



Executing 1000 random concurrent queries on nine nodes.

BIFOLD The Berlin Institute for the Foundations of Learning and Data
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
ELEGANT
exdra
SPP2037 SCALABLE DATA MANAGEMENT FOR FUTURE HARDWARE
www.bifold.berlin