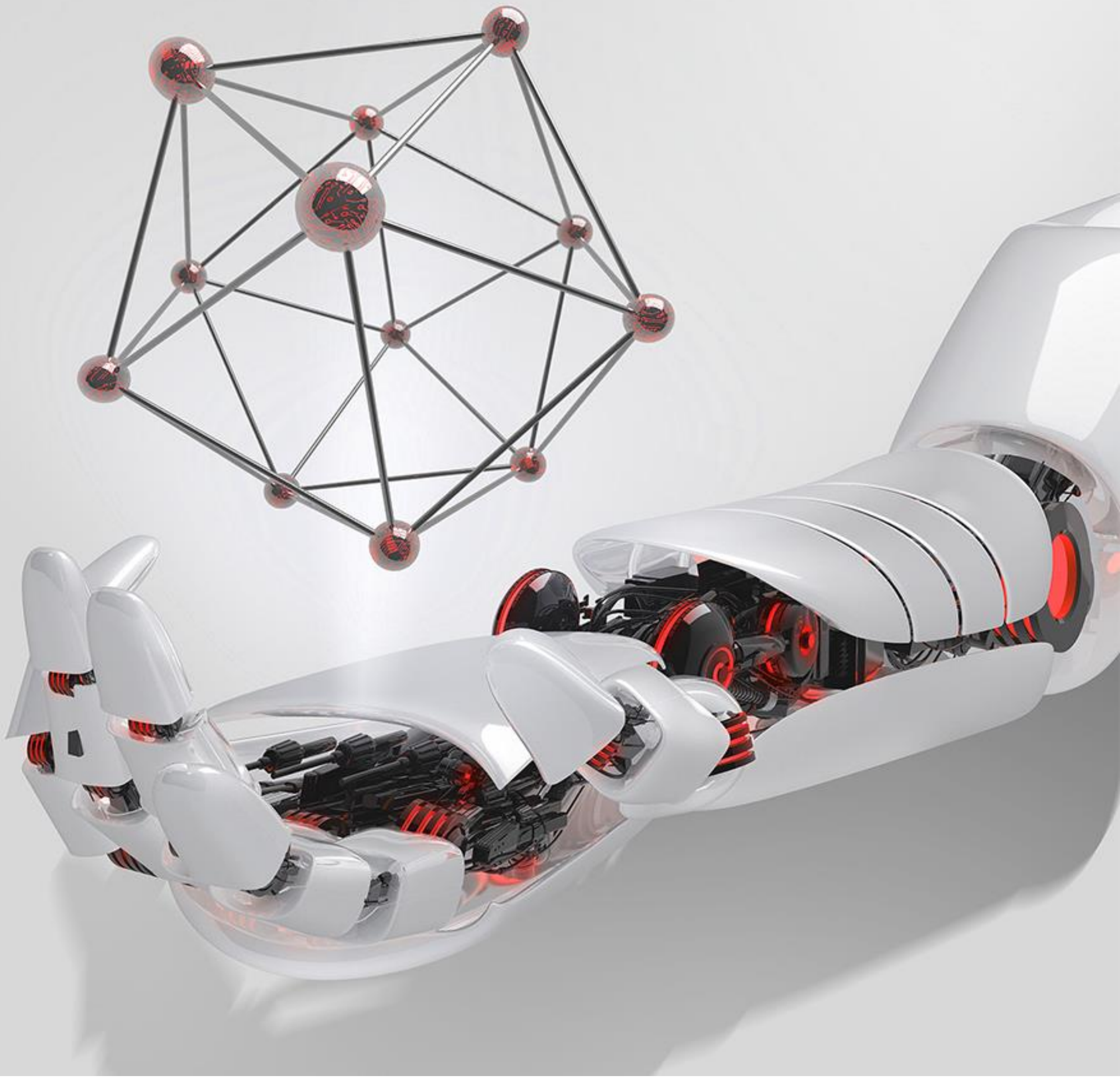
17 / June / 2019

Big Data and online streaming machine learning

Big Data, Fast Data, All Data



Dr. Cristian Axenie

Senior Research Engineer AI and ML

AI and ML in Big Data Team

IT Software Infrastructure Dept.

Security Level: **Open**

HUAWEI European Research Center

HUAWEI

Introducing the speaker

TUM PhD in Neuroscience and Robotics, Summa cum Laude

Specialized in designing and implementing AI and ML system for real-world problems

Academic Research

Head of Research Lab
AI and VR
As of 2017

AUDI KONFUZIUS-INSTITUT INGOLSTADT

Lecturer
As of 2017

Technische Hochschule Ingolstadt

Postdoctoral Fellow, Lecturer
2016-2017

TUM bccn munich Center of Competence on Neuroengineering

Research Assistant (PhD)
2011-2016

Technical University of Munich TUM

Industry Research

Senior Research Engineer
AI, ML & Big Data
As of 2017

HUAWEI

Software Engineer
Automotive
2009-2011

intel

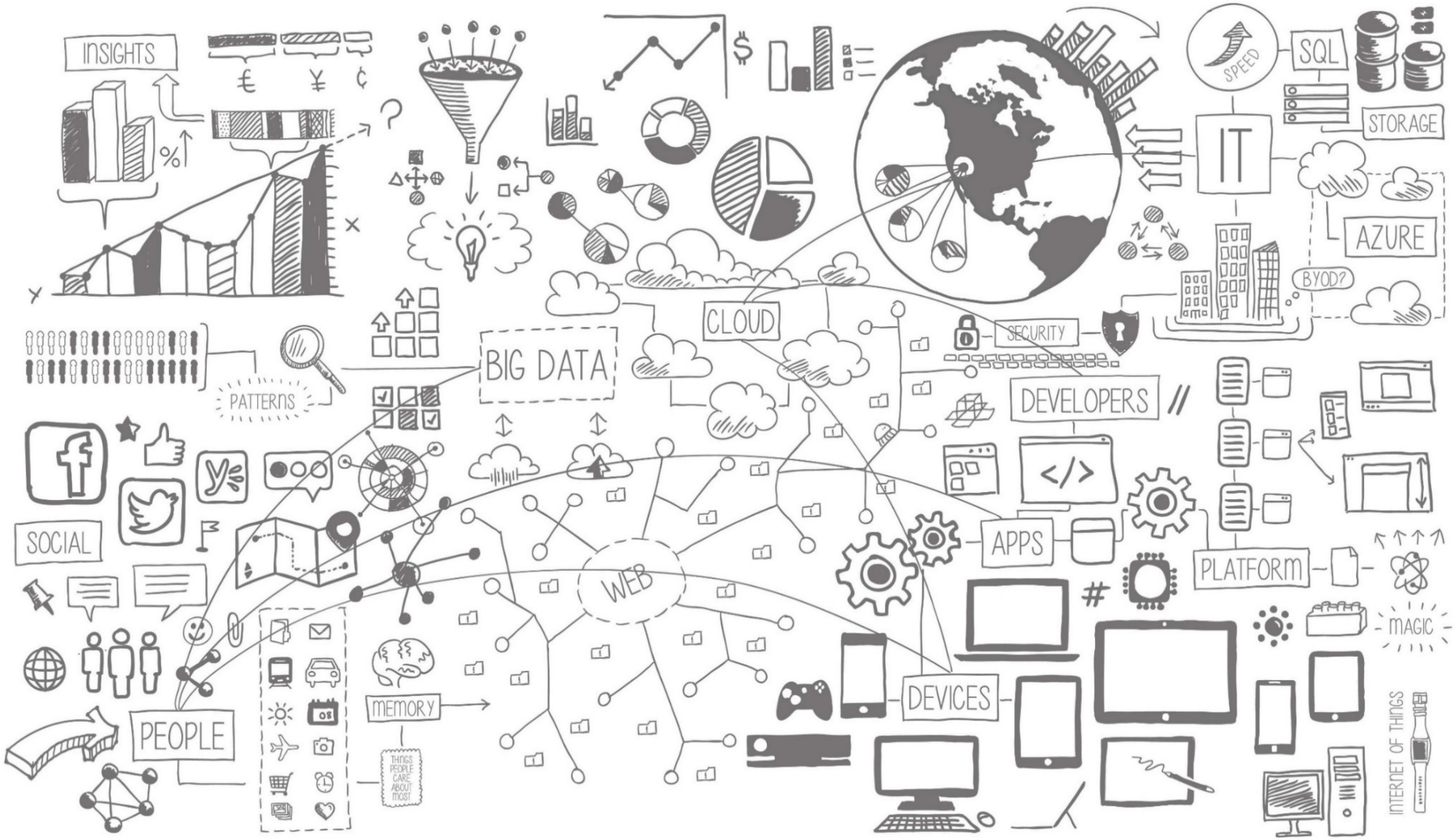
Software Engineer
Automotive
2009-2011

WIND RIVER

Software Engineer
Embedded Systems
2007-2008

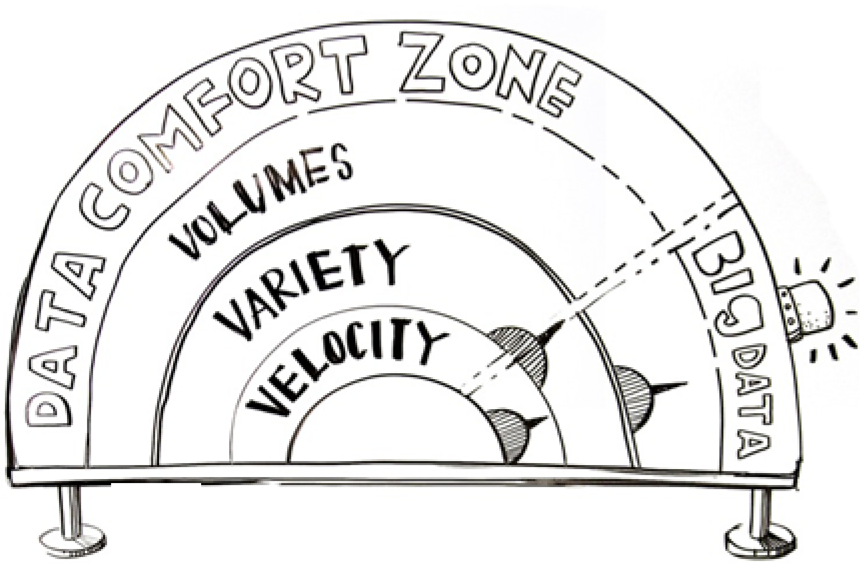
HUAWEI

(Dis)ambiguating Big Data



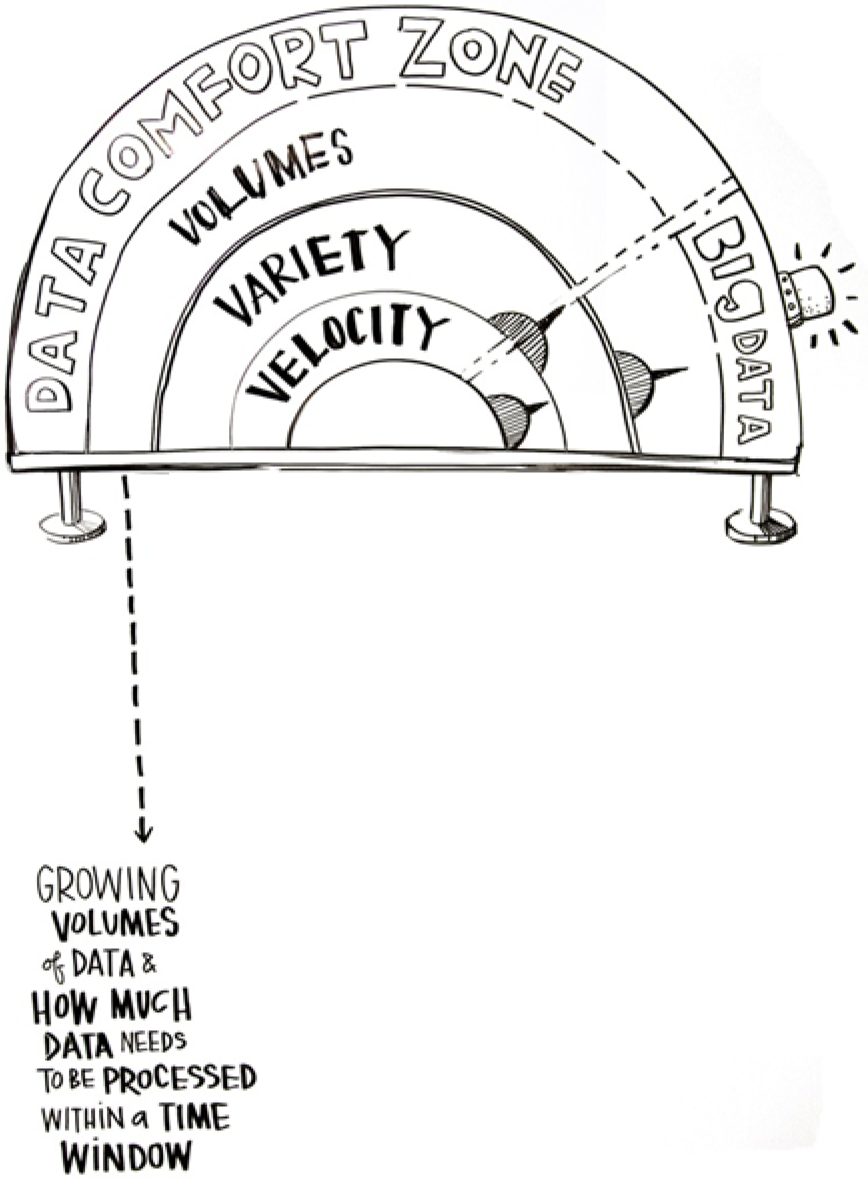
HUAWEI

Big Data in a Nutshell

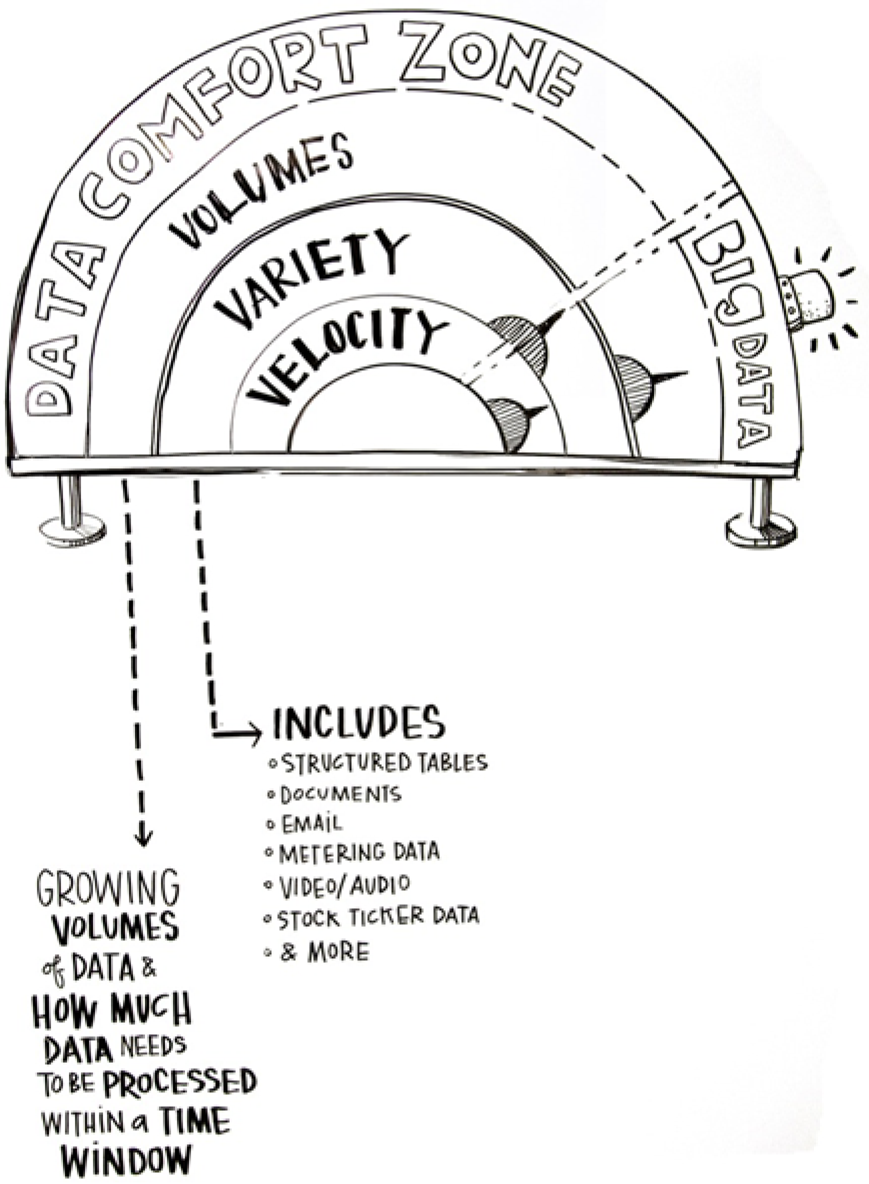


HUAWEI

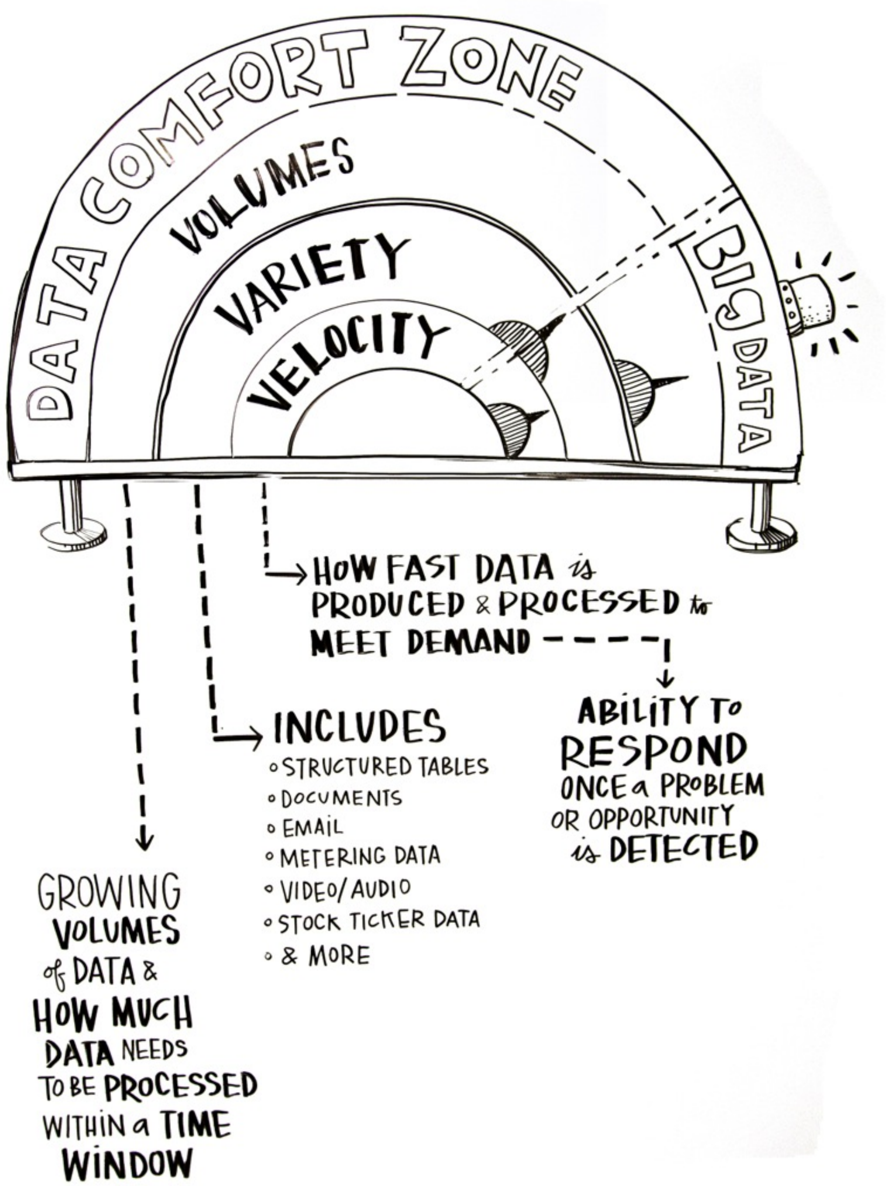
Big Data in a Nutshell



Big Data in a Nutshell



Big Data in a Nutshell



Big Data Stream Processing: A gentle introduction

Big Data Stream Processing: A gentle introduction

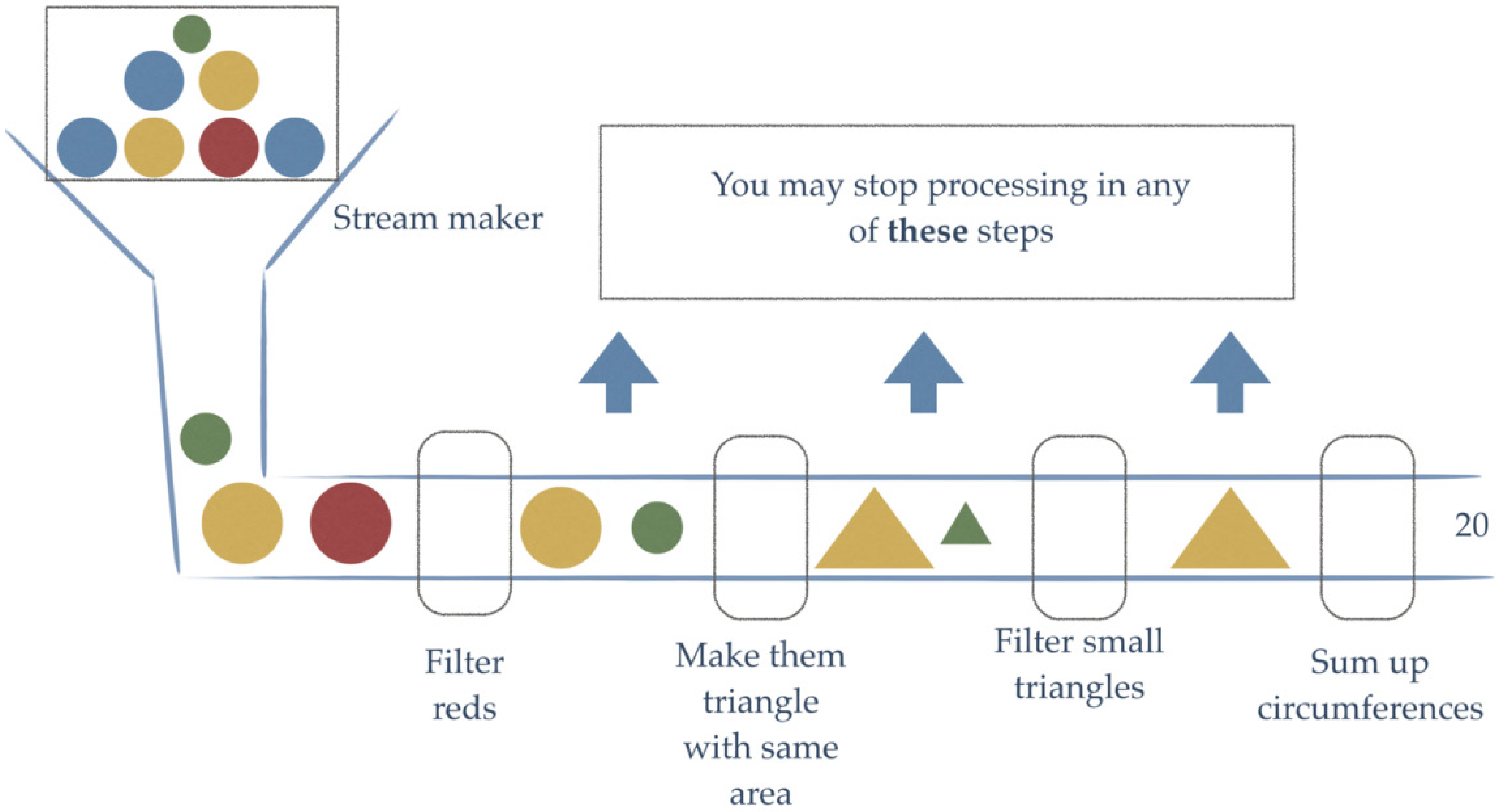
Stream processing paradigm simplifies parallel software and hardware by restricting the parallel computation that can be performed.

Given a sequence of data (***a stream***), a series of operations (***functions***) is applied to each element in the stream, in a declarative way, we specify **what** we want to achieve and **not how**.

Big Data Stream Processing: A gentle introduction

Stream processing paradigm simplifies parallel software and hardware by restricting the parallel computation that can be performed.

Given a sequence of data (***a stream***), a series of operations (***functions***) is applied to each element in the stream, in a declarative way, we specify **what** we want to achieve and **not how**.



HUAWEI

Big Data Stream Learning: Why is it different?

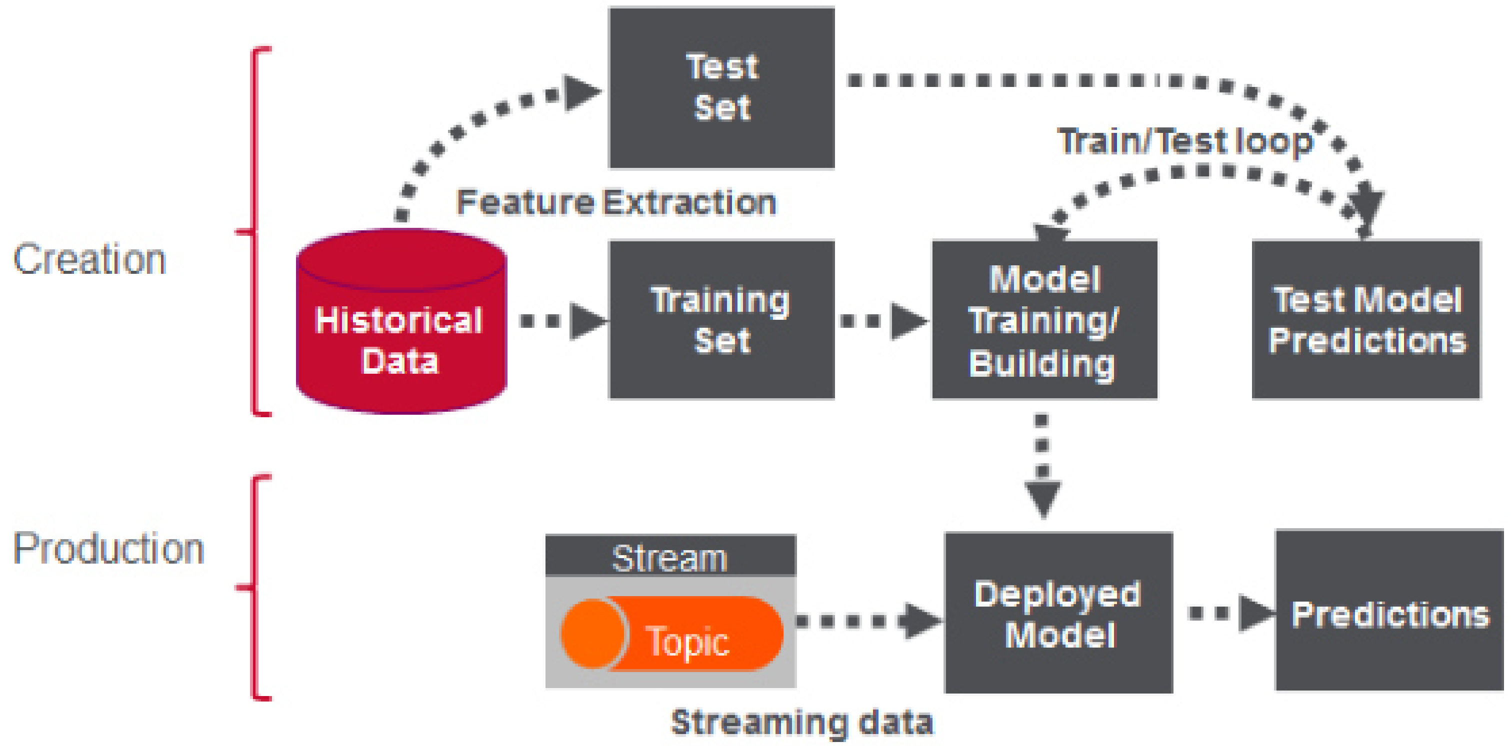
Big Data Stream Learning: Why is it different?

Big Data Stream Learning is more challenging than **batch** or **offline learning**, since the data may **not preserve the same distribution** over the lifetime of the stream.
Moreover, each example coming in a stream can only be **processed once**, or needs to be summarized with a **small memory footprint**, and the learning algorithms must be **efficient**.

HUAWEI

Big Data Stream Learning: Why is it different?

Big Data Stream Learning is more challenging than **batch** or **offline learning**, since the data may **not preserve the same distribution** over the lifetime of the stream.
Moreover, each example coming in a stream can only be **processed once**, or needs to be summarized with a **small memory footprint**, and the learning algorithms must be **efficient**.



Big Data Stream Learning: Where’ s the catch?

Big Data Stream Learning: Where’s the catch?

In order to deal with **evolving data streams**, the model learnt from the streaming data must **capture up-to-date trends and transient patterns** in the stream.

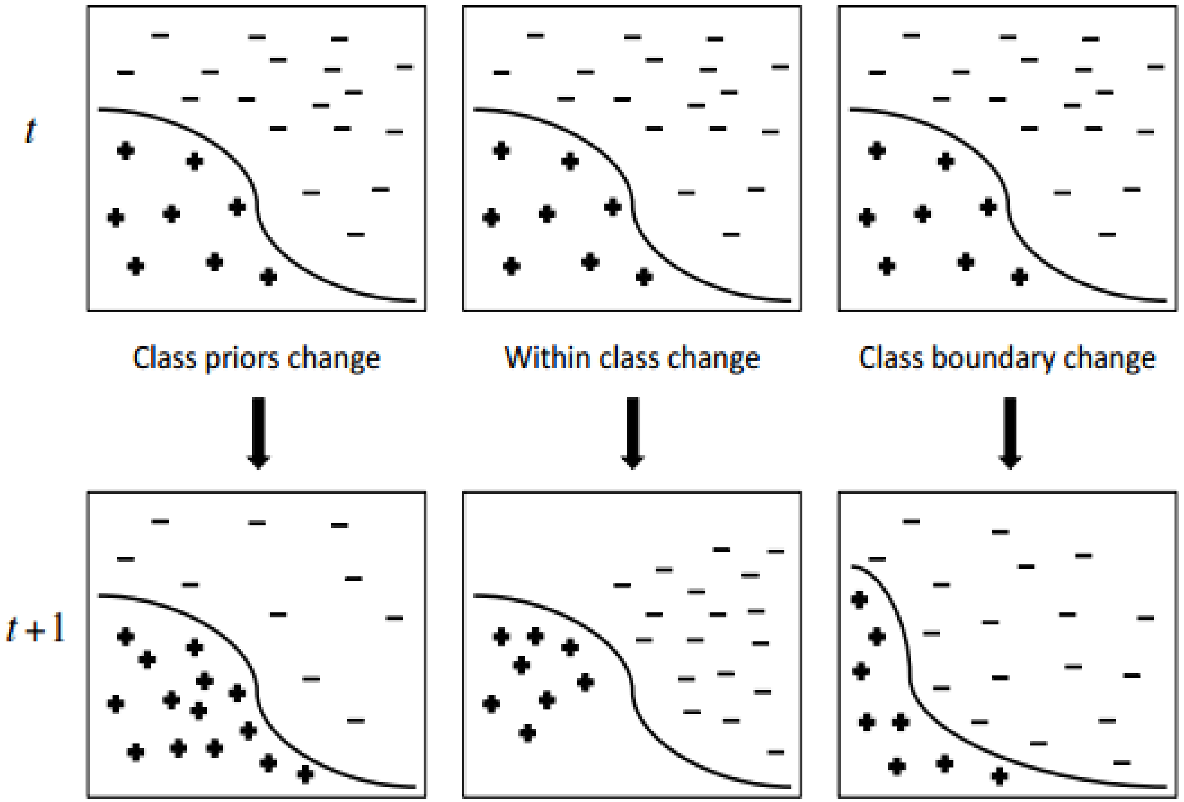
Updating the model by incorporating **new examples**, we must also **eliminate the effects of outdated examples** representing outdated concepts through **one-pass**.

HUAWEI

Big Data Stream Learning: Where’ s the catch?

In order to deal with **evolving data streams**, the model learnt from the streaming data must **capture up-to-date trends and transient patterns** in the stream.

Updating the model by incorporating **new examples**, we must also **eliminate the effects of outdated examples** representing outdated concepts through **one-pass**.



Streaming Machine Learning Basics

Elements of Streaming Machine Learning

HUAWEI

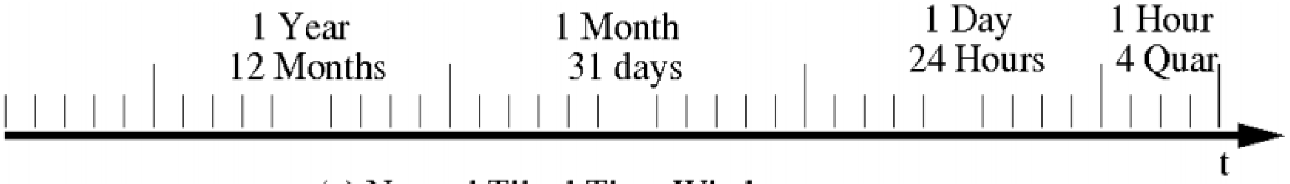
Elements of Streaming Machine Learning

Most strategies use variations of the **sliding window technique**: a window is maintained that keeps the most **recently read examples**, and from which **older examples are dropped** according to some **set of rules**.

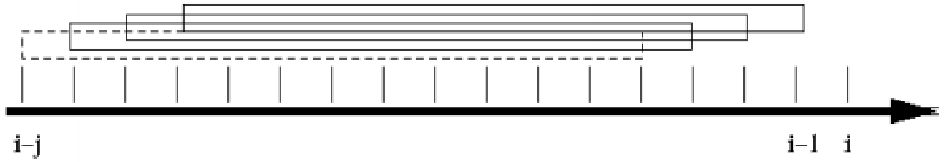
Elements of Streaming Machine Learning

Most strategies use variations of the **sliding window technique**: a window is maintained that keeps the most **recently read examples**, and from which **older examples are dropped** according to some **set of rules**.

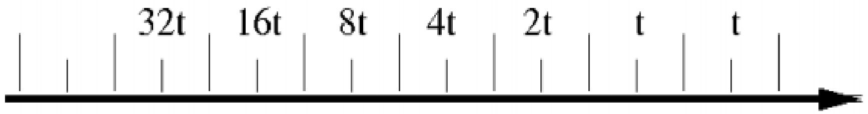
(a) Landmark Window



(a) Natural Tilted Time Window



(b) Sliding Window



b) Logarithmic Tilted Time Window

Elements of Streaming Machine Learning

The contents of the **sliding window** can be used for the three tasks:

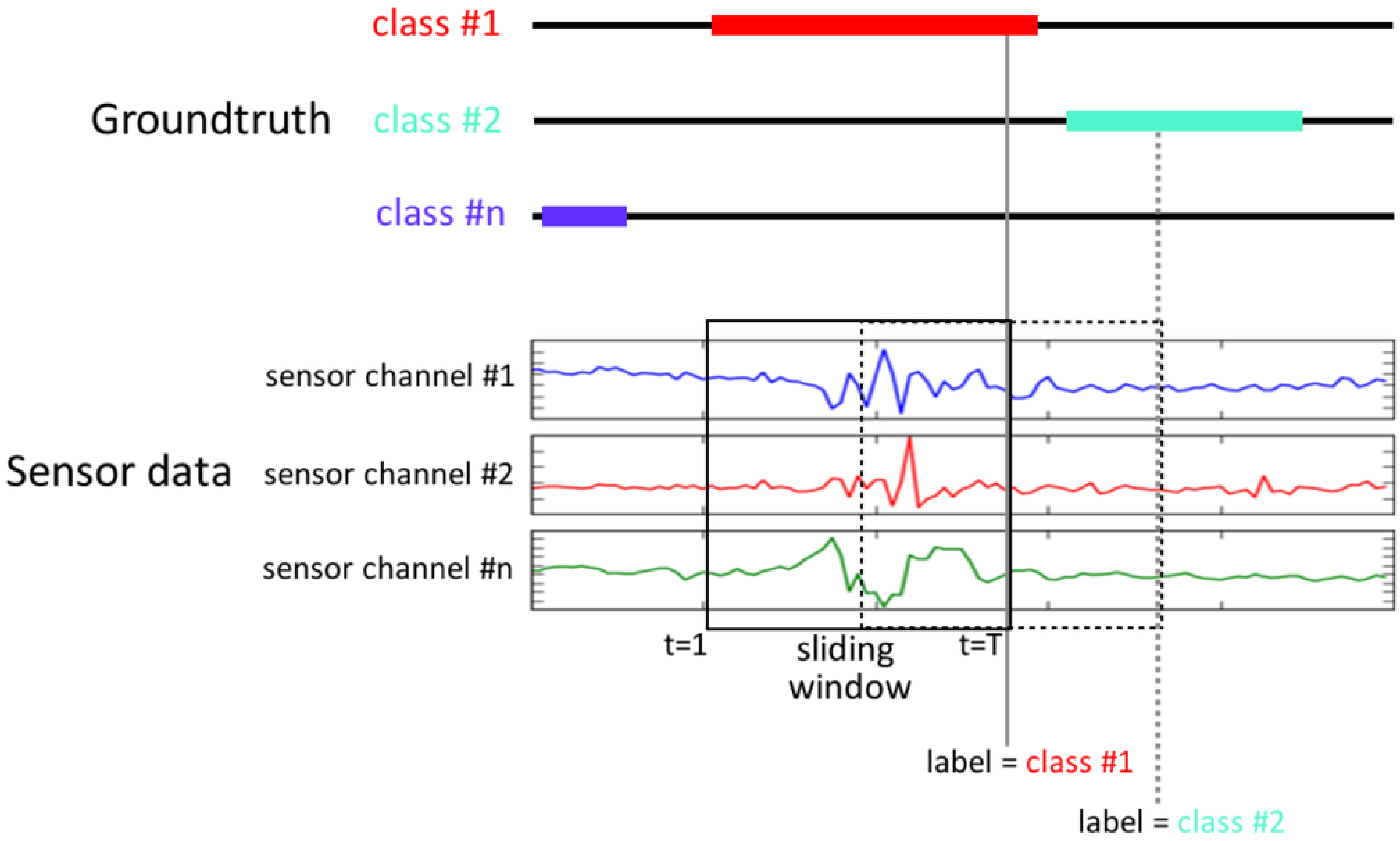
1) to **detect change** (e.g., by using some statistical test on different sub-windows),
2) to obtain **updated statistics / criteria** from the **recent examples**, and
3) to have **data** to **rebuild** or **update the model** after **data has changed**.

HUAWEI

Elements of Streaming Machine Learning

The contents of the **sliding window** can be used for the three tasks:

1) to **detect change** (e.g., by using some statistical test on different sub-windows),
2) to obtain **updated statistics / criteria** from the **recent examples**, and
3) to have **data** to **rebuild** or **update the model** after **data has changed**.



HUAWEI

Elements of Streaming Machine Learning

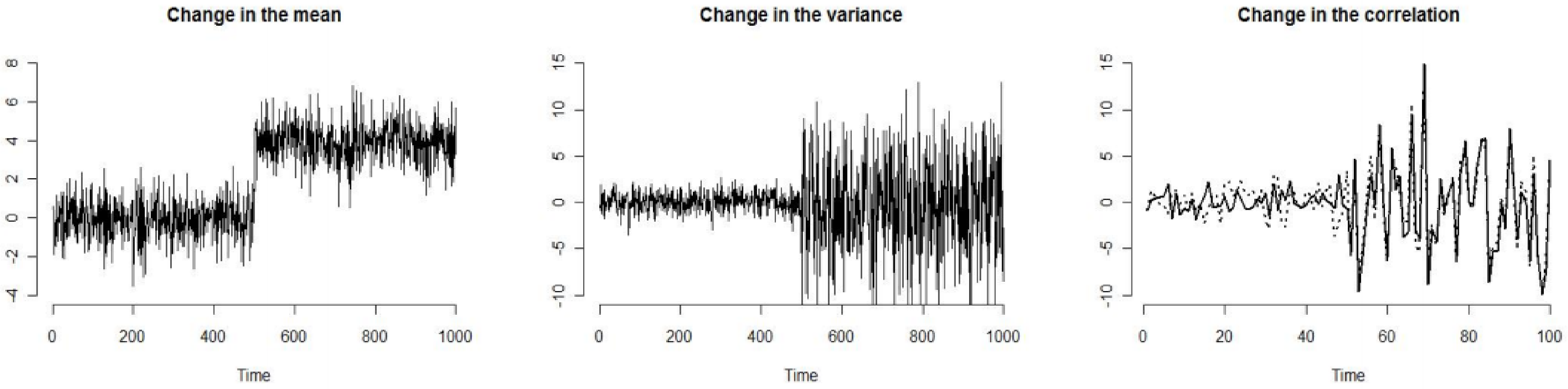
Normally, the user is caught in a **tradeoff without solution**:

- a **small size** (so that the **window reflects** accurately the current **distribution**)
- a **large size** (so that many **examples** are available to work on, **increasing accuracy** in periods of stability).

Elements of Streaming Machine Learning

Normally, the user is caught in a **tradeoff without solution**:

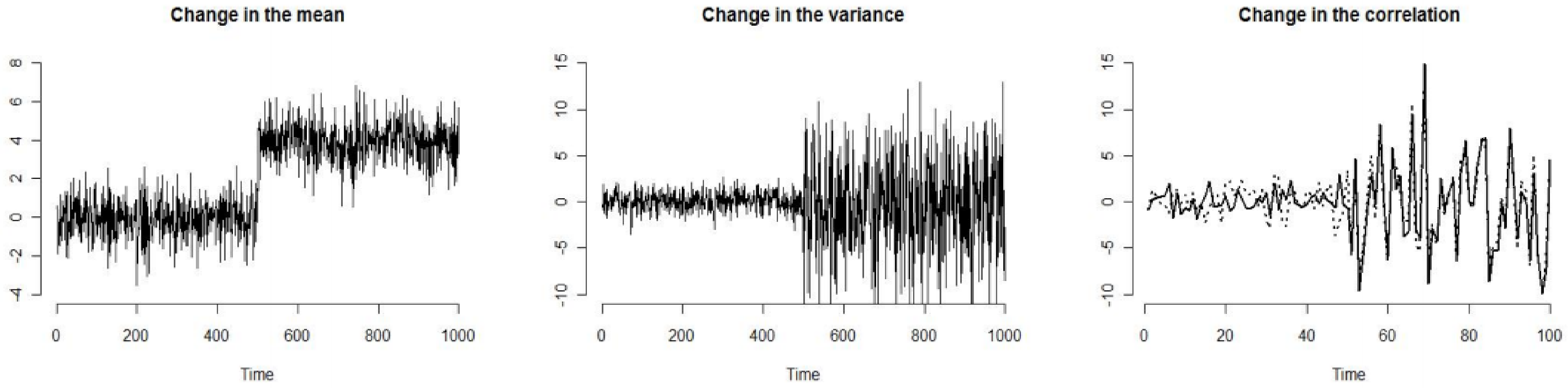
- a **small size** (so that the **window reflects** accurately the current **distribution**)
- a **large size** (so that many **examples** are available to work on, **increasing accuracy** in periods of stability).



Elements of Streaming Machine Learning

Normally, the user is caught in a **tradeoff without solution**:

- a **small size** (so that the **window reflects** accurately the current **distribution**)
- a **large size** (so that many **examples** are available to work on, **increasing accuracy** in periods of stability).



Currently, it has been proposed to use **windows** of **variable size**.

HUAWEI

Streaming Machine Learning in Action

Streaming Machine Learning I

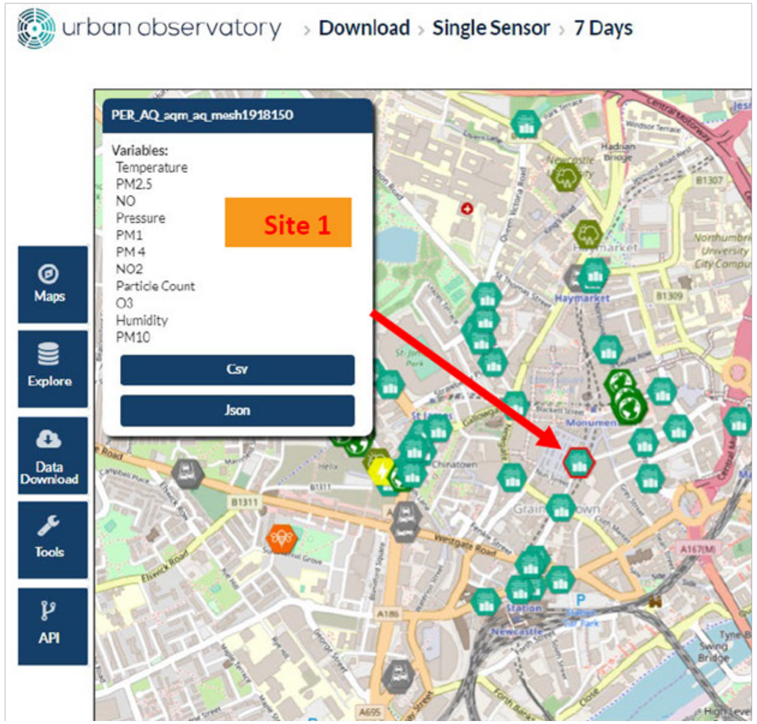
Example use-cases **Pollution GO**

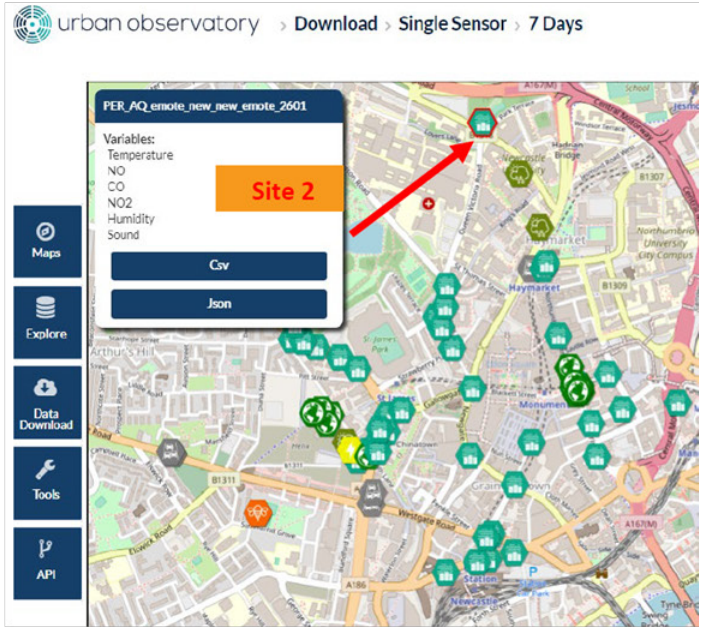
Urban Observatory project at **Newcastle University:**

- request sensory data to tackle specific challenges, such as flooding or air quality within **Newcastle Metropolitan** area: http://uoweb1.ncl.ac.uk/

- assuming we interrogate **two sites** (locations) in the **Newcastle Urban Observatory**, marked as red hexagons in the images.

You can see what is the **available sensory information** at **each site**.





HUAWEI

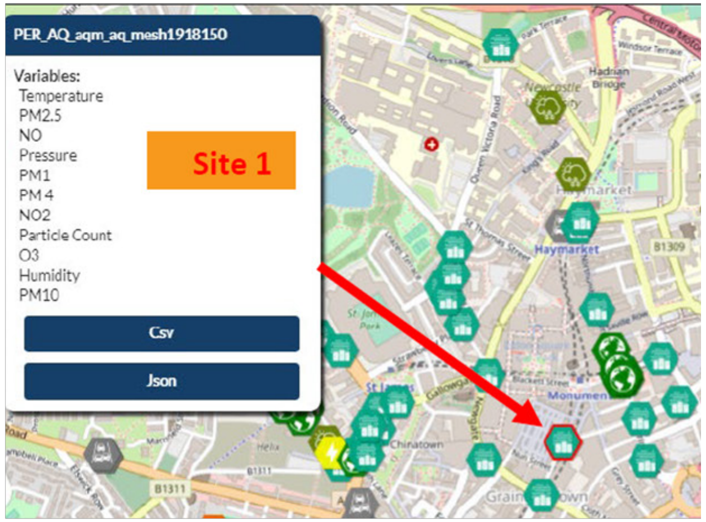
Streaming Machine Learning I

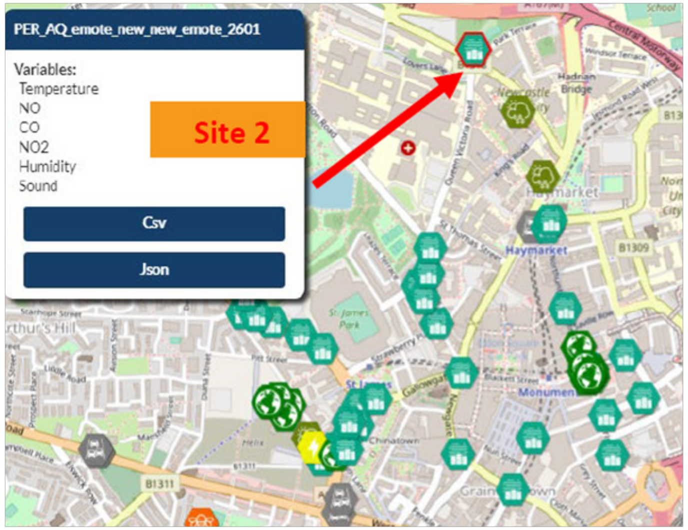
Example use-cases **Pollution GO**

The task is to **learn pair-wise correlations** among the **available sensors**.

The **system performs unsupervised learning of functional relationships** between two **input sensory streams** (e.g. NO in Site1 and Temperature in Site 1).

This **neural network** based system can be employed in **various solutions** as a **tool to learn pair-wise sensory correlations** among **sensors within / between spatial locations** (e.g. CO in Site 2 and O3 in Site 1).





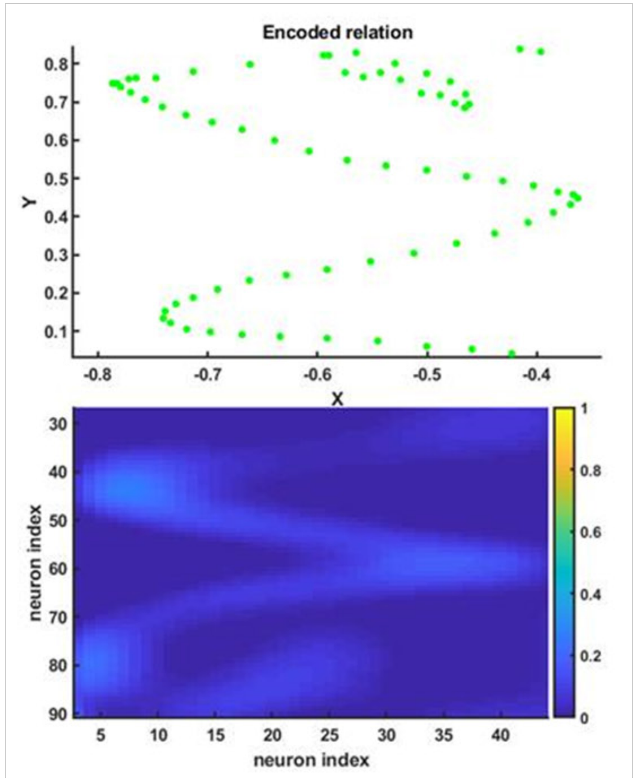
HUAWEI

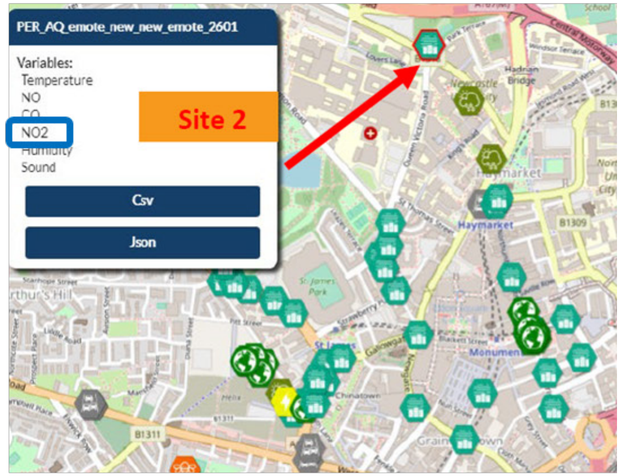
Streaming Machine Learning I

Example use-cases **Pollution GO**

The **power of the approach** is that one can **learn / extract sensory correlations** in **various constellations**:

- Learn **between location within sensor** correlations
  - example **NO2** for **site 1 and site 2**, corresponding to X and Y respectively, over a range





HUAWEI

Streaming Machine Learning II

Example use-cases **Traffic GO**

The **approach** can **learn / extract sensory correlations** in **various scenarios**.

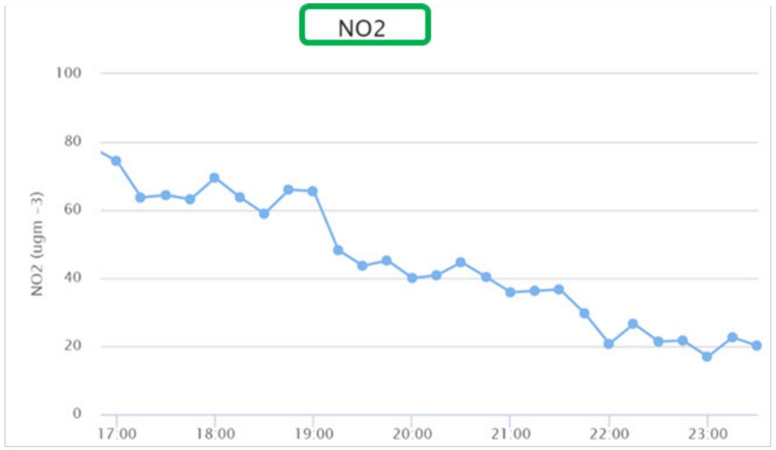
In the traffic scenario we propose to **learn the correlation** between **Environment parameters** (NO, O3, NO2, NOx), **Weather** (Humidity, Rain) and the **Traffic Flow** (number of vehicles) at a site.

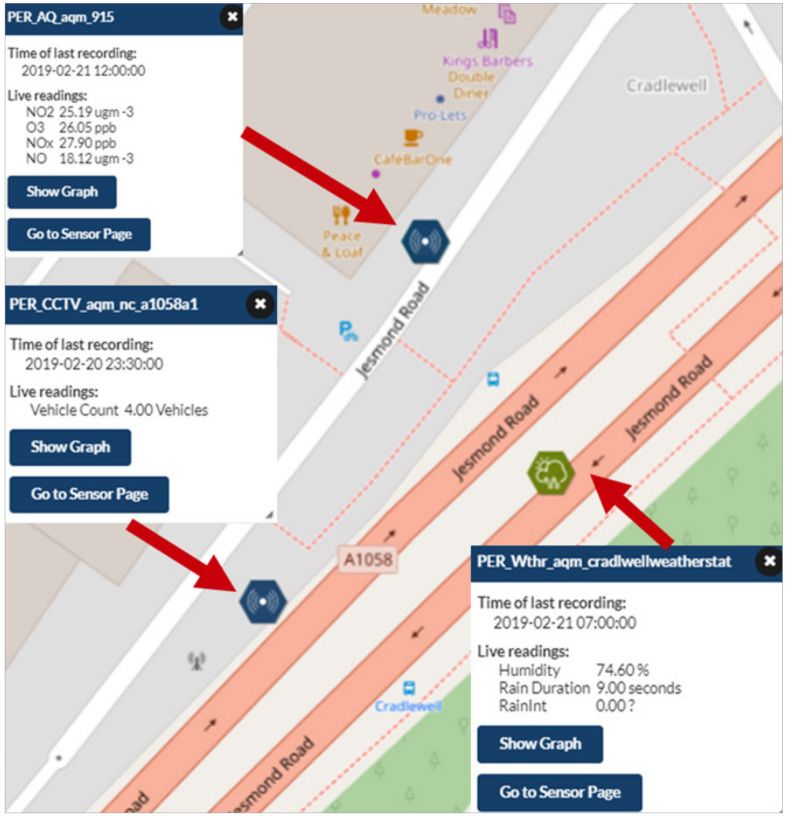
Once **learnt the correlation** we can use it to **infer traffic flow** in regions where **we do not have traffic sensors** installed but **all other sensors** are present.

HUAWEI

Streaming Machine Learning II

Example use-cases **Traffic GO**

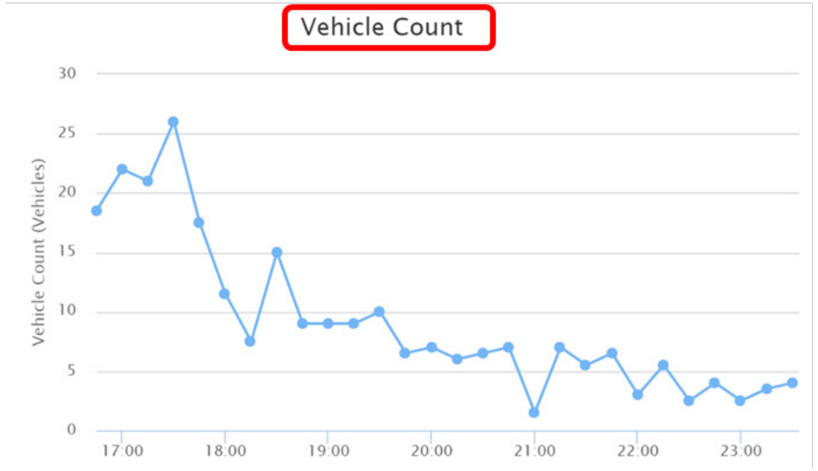


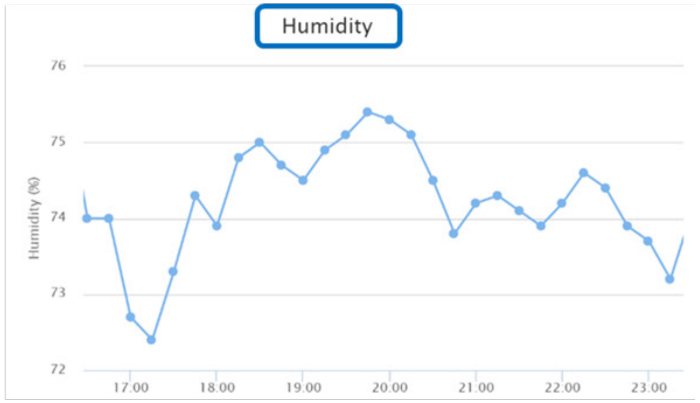


urban observatory

EPSRC UKCRIC Newcastle University

http://uoweb1.ncl.ac.uk/

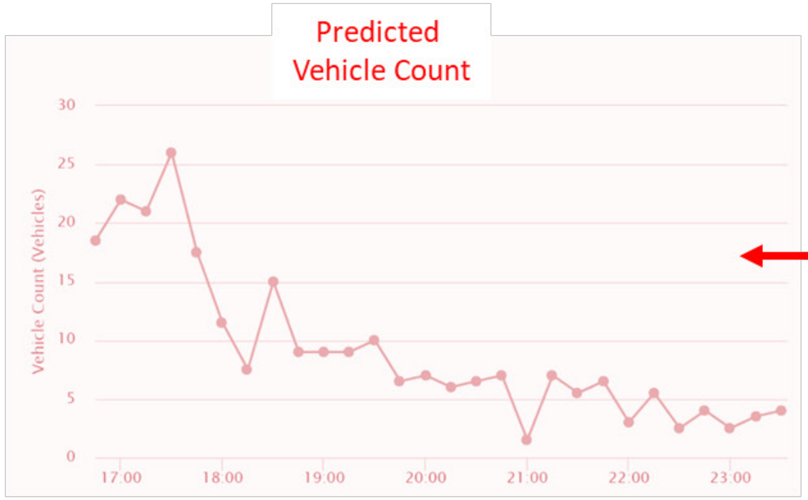


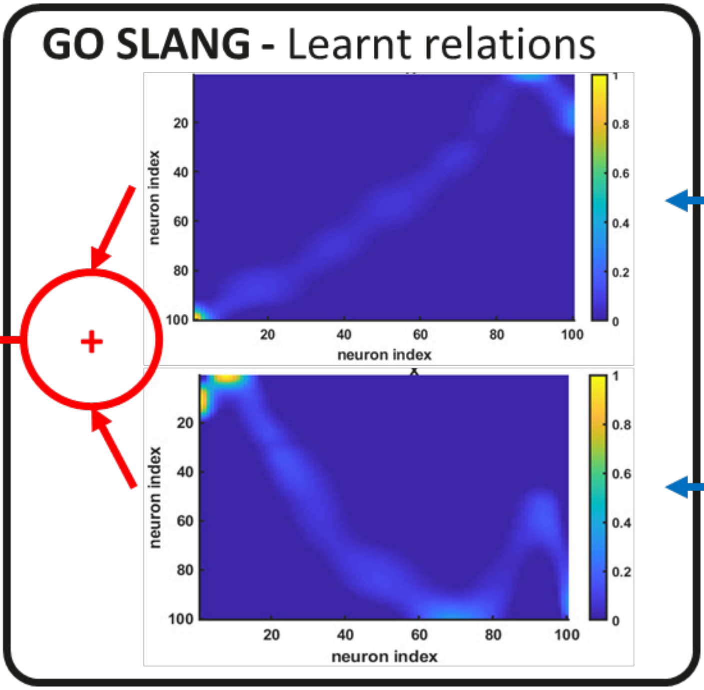


HUAWEI

Streaming Machine Learning II

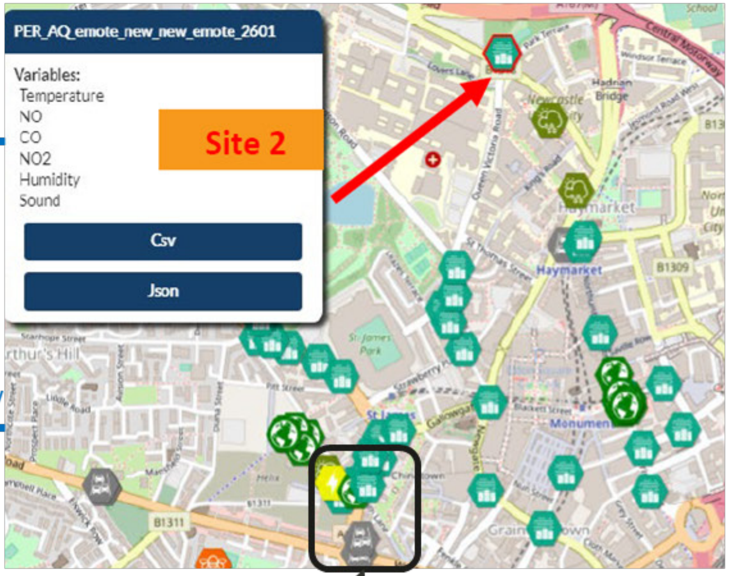
Once **learnt the correlation** we can use it to **infer traffic flow** in regions where **we do not have traffic sensors** installed but **all other sensors** are present (e.g. Humidity and NO2).





NO2

Humidity



If the **traffic sensor is failing / defect**,
The system uses previously learnt relations to infer a plausible prediction.

HUAWEI

Streaming Machine Learning III

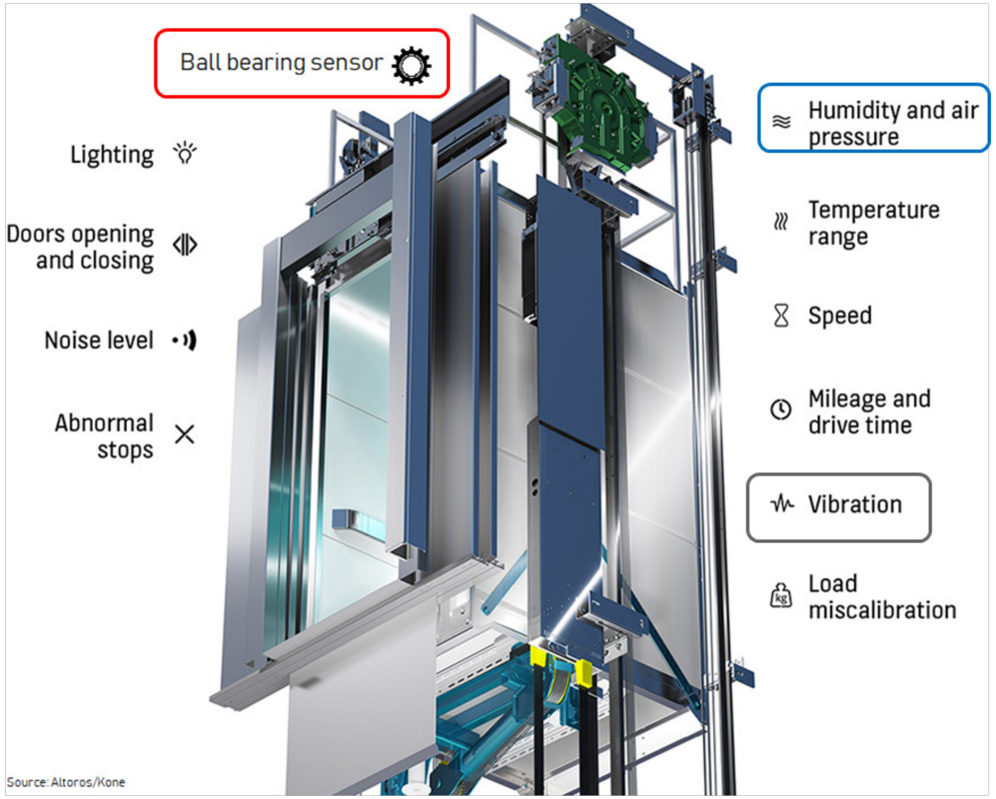
Example use-case Elevator Analytics

The **approach** can **learn / extract sensory correlations** for Predictive Maintenance of Elevator Doors.

For an **elevator car door** the system can **learn the correlation** between:

- **Electromechanical sensors** (Door Ball Bearing Sensor)
- **Ambiance**(Humidity)
- **Physics**(Vibration)

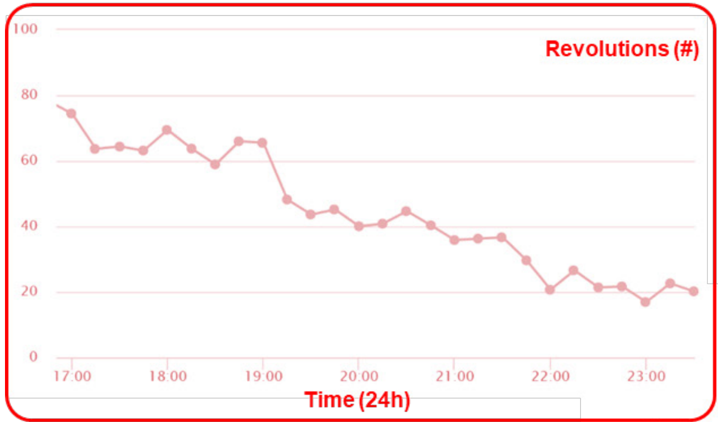
Once **learnt the correlation in operational settings** we can use it to **infer anomalous** operation of the doors.

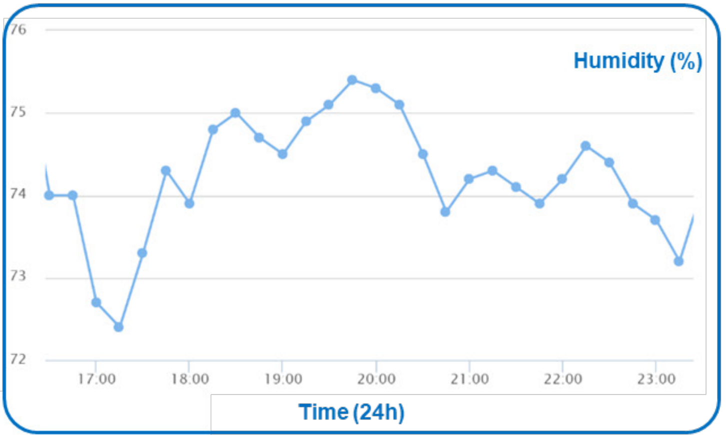


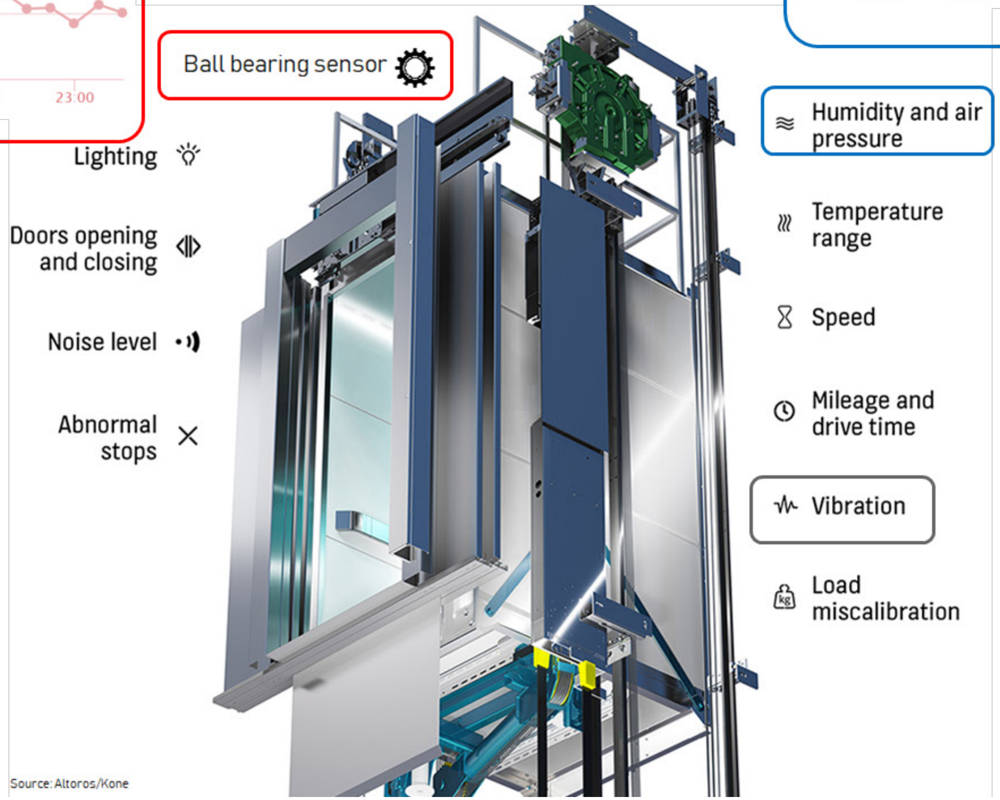
HUAWEI

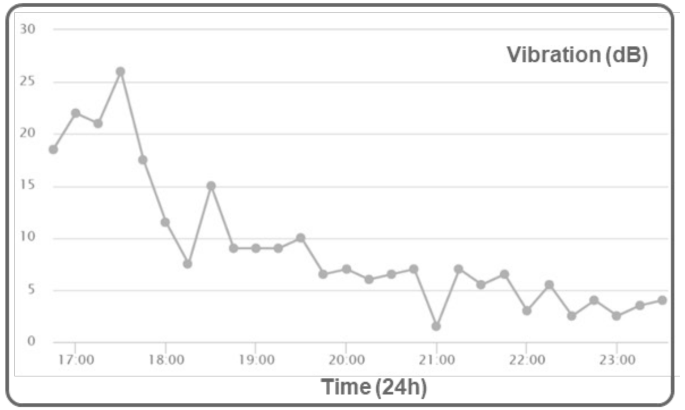
Streaming Machine Learning III

Example use-case **Elevator Analytics**







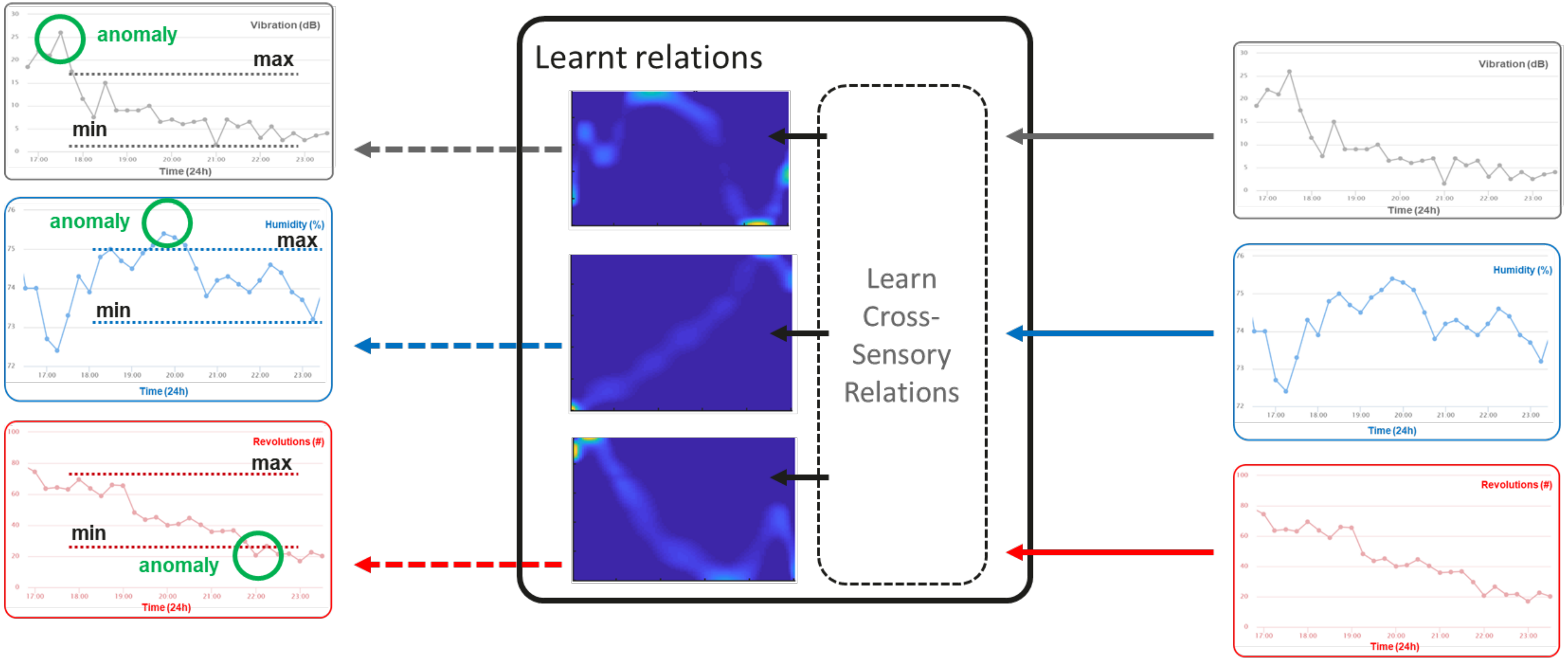


We have the operation data, timeseries sampled at 10Hz in high-peak and evening elevator usage in a building (between 16:30 and 23:30).

Streaming Machine Learning III

Example use-case **Elevator Analytics**

Once **learnt the correlation** we can use it to **trigger alarms for anomalies / outliers**.



HUAWEI

Conclusions

Conclusions

Induced by ubiquitous scenarios *finite training sets*, *static models*, and *stationary distributions* must be completely redefined.

Conclusions

Induced by ubiquitous scenarios *finite training sets*, *static models*, and *stationary distributions* must be completely redefined.

The characteristics of the streaming data entail a new vision due to the fact that:

Conclusions

Induced by ubiquitous scenarios *finite training sets*, *static models*, and *stationary distributions* must be completely redefined.

The characteristics of the streaming data entail a new vision due to the fact that:

- Data are made available through **unlimited streams** that **continuously flow**, eventually at **high speed**, over time;

Conclusions

Induced by ubiquitous scenarios *finite training sets*, *static models*, and *stationary distributions* must be completely redefined.

The characteristics of the streaming data entail a new vision due to the fact that:

- Data are made available through **unlimited streams** that **continuously flow**, eventually at **high speed**, over time;
- The **underlying regularities may evolve** over time rather than being stationary;

Conclusions

Induced by ubiquitous scenarios *finite training sets*, *static models*, and *stationary distributions* must be completely redefined.

The characteristics of the streaming data entail a new vision due to the fact that:

- Data are made available through **unlimited streams** that **continuously flow**, eventually at **high speed**, over time;
- The **underlying regularities may evolve** over time rather than being stationary;
- The data can **no longer** be considered as **independent and identically distributed**;

Conclusions

Induced by ubiquitous scenarios *finite training sets*, *static models*, and *stationary distributions* must be completely redefined.

The characteristics of the streaming data entail a new vision due to the fact that:

- Data are made available through **unlimited streams** that **continuously flow**, eventually at **high speed**, over time;
- The **underlying regularities may evolve** over time rather than being stationary;
- The data can **no longer** be considered as **independent and identically distributed**;
- The **data** are now often **spatially** as well as **time situated**.

Conclusions

Induced by ubiquitous scenarios *finite training sets*, *static models*, and *stationary distributions* must be completely redefined.

The characteristics of the streaming data entail a new vision due to the fact that:

- Data are made available through **unlimited streams** that **continuously flow**, eventually at **high speed**, over time;
- The **underlying regularities may evolve** over time rather than being stationary;
- The data can **no longer** be considered as **independent and identically distributed**;
- The **data** are now often **spatially** as well as **time situated**.

A new era of machine learning?

HUAWEI

Thank you.

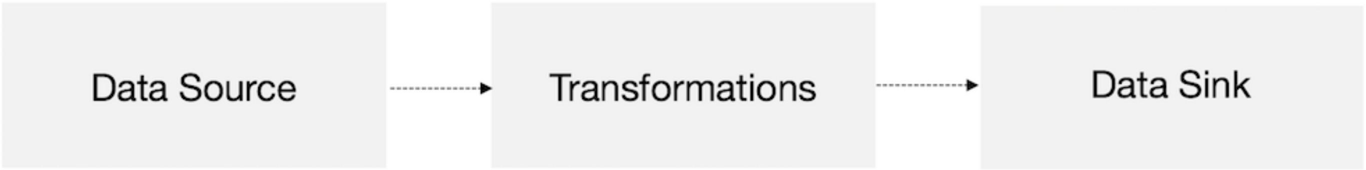
Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI

Stream Processing Engines

Stream Processing Engines

Flink



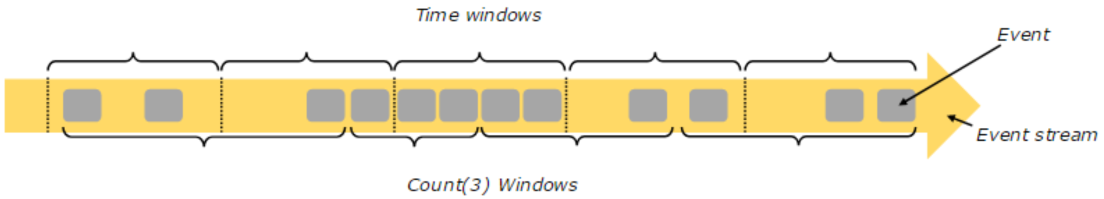
HUAWEI

Stream Processing Engines

Flink

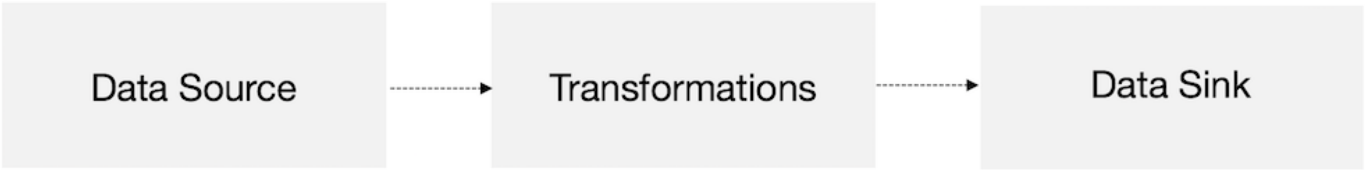
Data Source → Transformations → Data Sink

Aggregating events (e.g., counts, sums) works differently on streams because it is **impossible to count** all (unbounded).

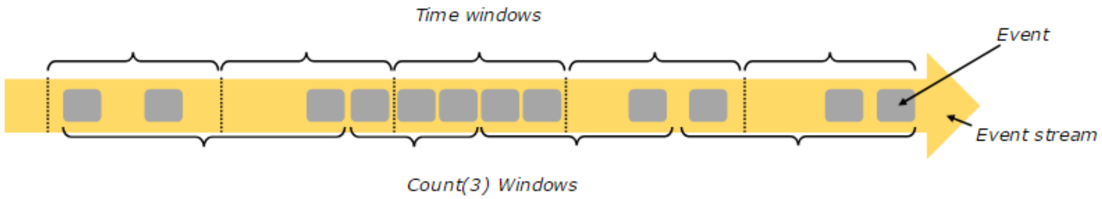


Stream Processing Engines

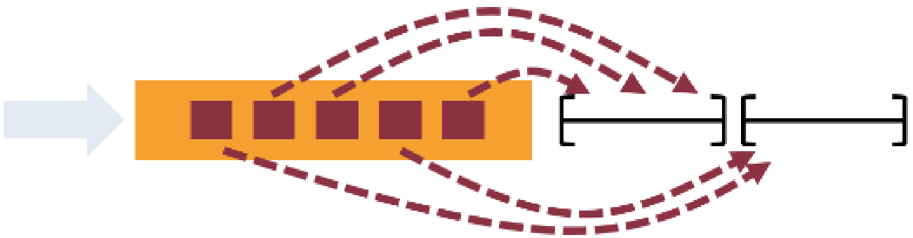
Flink



Aggregating events (e.g., counts, sums) works differently on streams because it is **impossible to count** all (unbounded).

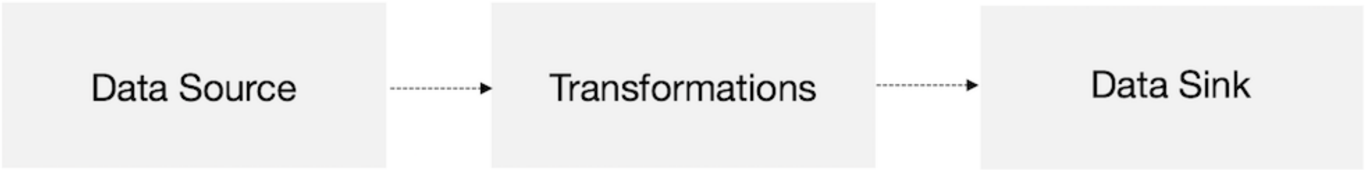


Stream processing and **windowing** makes it easy to compute accurate results over streams where **events** arrive **out of order** and where **events may arrive delayed**.

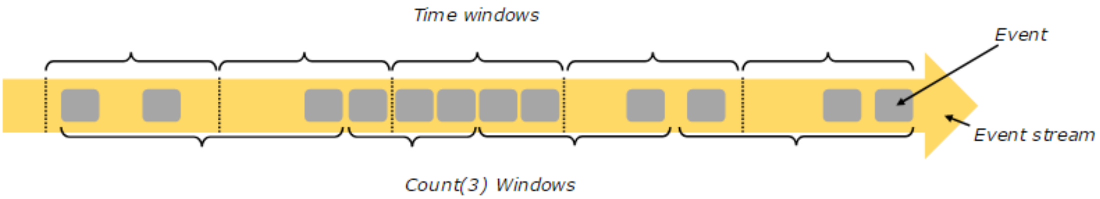


Stream Processing Engines

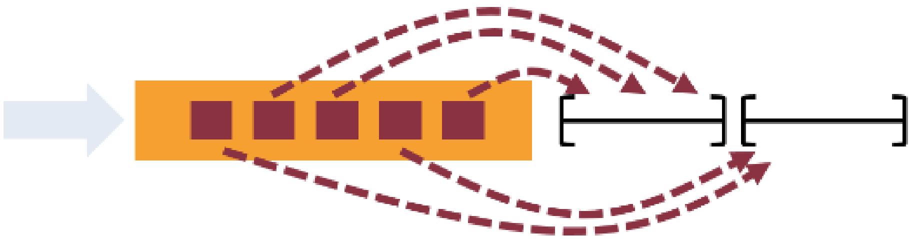
Flink



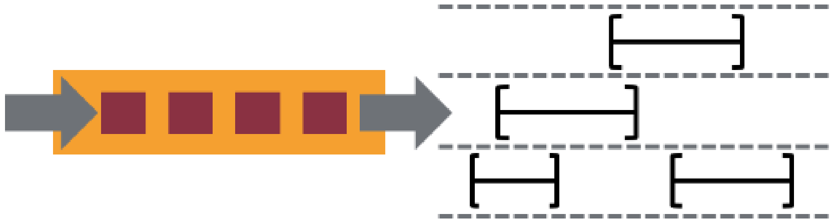
Aggregating events (e.g., counts, sums) works differently on streams because it is **impossible to count** all (unbounded).



Stream processing and **windowing** makes it easy to compute accurate results over streams where **events** arrive **out of order** and where **events may arrive delayed**.



Windowing based on time, count, and data-driven windows. Windows can be customized with **flexible triggering** conditions to support **sophisticated streaming patterns**.



Stream Processing Engines

Flink

When executed, Flink programs are mapped to **streaming dataflows**, consisting of **streams** and **transformation operators**.

Each **dataflow** starts with one or more **sources** and ends in one or more **sinks**.

The dataflows resemble arbitrary directed acyclic graphs (DAGs).

Stream Processing Engines

Flink

When executed, Flink programs are mapped to **streaming dataflows**, consisting of **streams** and **transformation operators**.

Each **dataflow** starts with one or more **sources** and ends in one or more **sinks**.

The dataflows resemble arbitrary directed acyclic graphs (DAGs).

```
DataStream<String> lines = env.addSource(
                              new FlinkKafkaConsumer<>(…));        } Source

DataStream<Event> events = lines.map((line) -> parse(line));       } Transformation

DataStream<Statistics> stats = events
        .keyBy("id")
        .timeWindow(Time.seconds(10))                               } Transformation
        .apply(new MyWindowAggregationFunction());

stats.addSink(new RollingSink(path));                               } Sink
```

HUAWEI

Stream Processing Engines

Flink

When executed, Flink programs are mapped to **streaming dataflows**, consisting of **streams** and **transformation operators**.

Each **dataflow** starts with one or more **sources** and ends in one or more **sinks**.

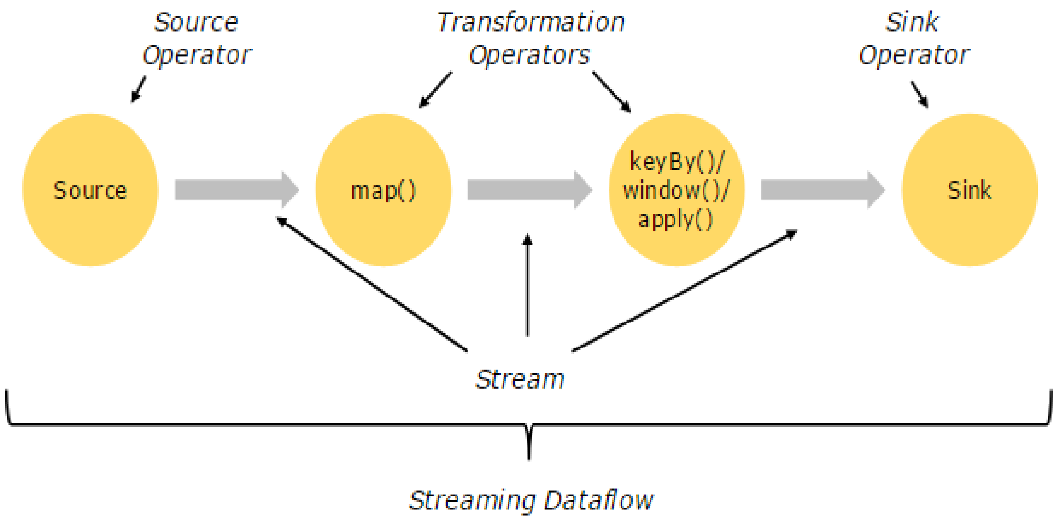
The dataflows resemble arbitrary directed acyclic graphs (DAGs).

```
DataStream<String> lines = env.addSource(                                Source
                         new FlinkKafkaConsumer<>(…));

DataStream<Event> events = lines.map((line) -> parse(line));             Transformation

DataStream<Statistics> stats = events
        .keyBy("id")
        .timeWindow(Time.seconds(10))                                    Transformation
        .apply(new MyWindowAggregationFunction());

stats.addSink(new RollingSink(path));                                    Sink
```



Stream Processing Engines

Flink

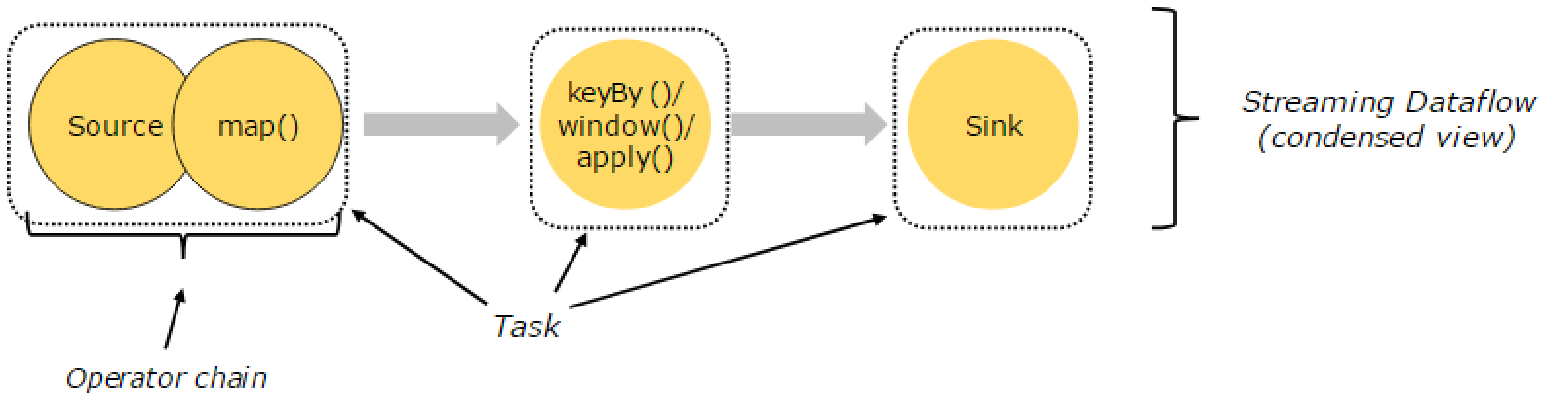
For distributed execution, Flink chains operator **subtasks** together into **tasks**. Each task is executed by one **thread**.

HUAWEI

Stream Processing Engines

Flink

For distributed execution, Flink chains operator **subtasks** together into **tasks**. Each task is executed by one **thread**.

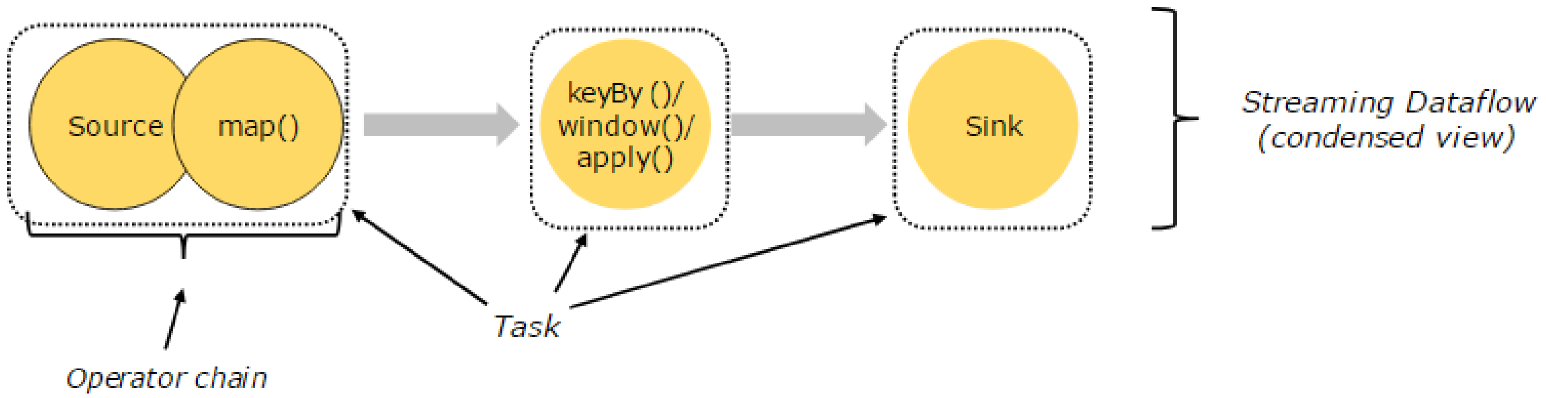


Stream Processing Engines

Flink

For distributed execution, Flink chains operator **subtasks** together into **tasks**. Each task is executed by one **thread**.

Chaining operators together into tasks is a useful optimization: it **reduces the overhead** of thread-to-thread handover and **buffering**, and **increases overall throughput** while **decreasing latency**.



HUAWEI

Stream Processing Engines

Flink

For distributed execution, Flink chains operator **subtasks** together into **tasks**. Each task is executed by one **thread**.

Chaining operators together into tasks is a useful optimization: it **reduces** the **overhead** of thread-to-thread handover and **buffering**, and **increases overall throughput** while **decreasing latency**.