

# ITEC874 — Big Data Technologies

## Week 11 Lecture 1: Analysing Streaming Data

Diego Mollá

Department of Computer Science  
Macquarie University

ITEC874 2019H2

# Programme

- 1 Data Streams
- 2 The Stream Model
- 3 Technologies for Stream Analytics

## Reading



# Programme

- 1 Data Streams
- 2 The Stream Model
- 3 Technologies for Stream Analytics

# Data Streams

## What is a data stream?

- A data stream is a sequence of data that are processed before the sequence ends.
- Data streams may be never-ending.

## Examples

**Image Data:** Surveillance cameras, satellite imagery, ...

**Sensor data:** Temperature, GPS coordinates, heart rate, ...

**Internet and Web Traffic :**

- Search queries;
- Posts from Twitter, Facebook, ...
- IP packets;
- Clicks.

# Applications I

## Mining query streams

Google wants to know what queries are more frequent today than yesterday.

## Mining click streams

Sydney Morning Herald wants to know which of its pages are getting an unusual number of hits in the past hour.

## Mining social network news feeds

E.g. A news agency looking for newsworthy topics on Twitter, Facebook.

# Applications II

## Sensor networks

Many sensors feeding into a central controller.

## Telephone call records

Data feeds into customer bills as well as settlements between telephone companies.

## IP packets monitored at a switch

- Gather information for optimal routing.
- Detect denial-of-service attacks.

# Data Streams as “Big Data”

The four “V’s” of Big Data applied to streams.

**Velocity:** Data may arrive faster than we can process it.

**Volume:** Accumulated data might not fit in the available storage space. We can think of data as **infinite**.

**Variety:** Data may change in time. Data that happened some time ago might not be relevant any more. We can think of data as **non-stationary**.

⇒ (This is not the standard meaning of variety...)

We still need to handle the “classic” issue of variety: we may need to handle multiple streams at once.

**Veracity:** Sensors may be faulty or temporarily down.

# Data Streams as “Big Data”

The four “V’s” of Big Data applied to streams.

**Velocity:** Data may arrive faster than we can process it.

**Volume:** Accumulated data might not fit in the available storage space. We can think of data as **infinite**.

**Variety:** Data may change in time. Data that happened some time ago might not be relevant any more. We can think of data as **non-stationary**.

⇒ (This is not the standard meaning of variety...)

We still need to handle the “classic” issue of variety: we may need to handle multiple streams at once.

**Veracity:** Sensors may be faulty or temporarily down.



# Data Streams as “Big Data”

The four “V’s” of Big Data applied to streams.

**Velocity:** Data may arrive faster than we can process it.

**Volume:** Accumulated data might not fit in the available storage space. We can think of data as **infinite**.

**Variety:** Data may change in time. Data that happened some time ago might not be relevant any more. We can think of data as **non-stationary**.

⇒ (This is not the standard meaning of variety...)

We still need to handle the “classic” issue of variety: we may need to handle multiple streams at once.

**Veracity:** Sensors may be faulty or temporarily down.

# Data Streams as “Big Data”

The four “V’s” of Big Data applied to streams.

**Velocity:** Data may arrive faster than we can process it.

**Volume:** Accumulated data might not fit in the available storage space. We can think of data as **infinite**.

**Variety:** Data may change in time. Data that happened some time ago might not be relevant any more. We can think of data as **non-stationary**.

⇒ (This is not the standard meaning of variety...)

We still need to handle the “classic” issue of variety: we may need to handle multiple streams at once.

**Veracity:** Sensors may be faulty or temporarily down.

# Data Streams as “Big Data”

The four “V’s” of Big Data applied to streams.

**Velocity:** Data may arrive faster than we can process it.

**Volume:** Accumulated data might not fit in the available storage space. We can think of data as **infinite**.

**Variety:** Data may change in time. Data that happened some time ago might not be relevant any more. We can think of data as **non-stationary**.

⇒ (This is not the standard meaning of variety...)

We still need to handle the “classic” issue of variety: we may need to handle multiple streams at once.

**Veracity:** Sensors may be faulty or temporarily down.

# Issues in Stream Processing

## Issues

**Velocity:** We may need to give up on processing all data.

**Volume:** We may need to build summaries.

- Not all ad-hoc questions can be answerable.

## Possible Solution

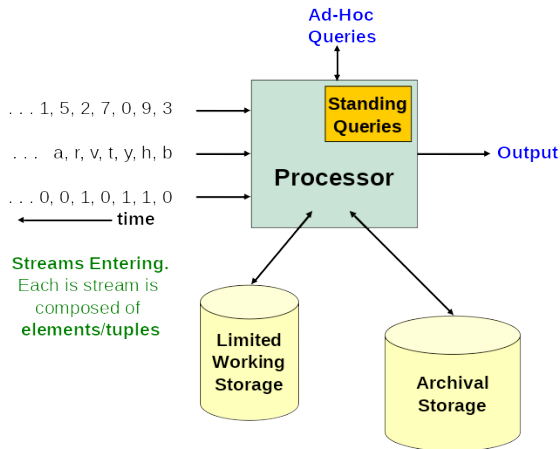
- Obtain an **approximate answer** to the question rather than an exact answer.
- For example, stream processing often focuses on the most recent data.

⇒ Focussing on recent data also addresses the issue of **variety**.

# Programme

- 1 Data Streams
- 2 The Stream Model
- 3 Technologies for Stream Analytics

# The Stream Model



<http://www.mmds.org/>

# Storage in the Stream Model

## Archival Storage

- Large storage for archival purposes.
- We assume it is not possible to answer queries from the archival store.
- Can be used only under special circumstances using time-consuming retrieval processes.

## Working Store

- Holds summaries or parts of streams.
- Can be used for answering queries.
- Might be in disk or in main memory.
- Cannot store all the data from all the streams.

# Storage in the Stream Model

## Archival Storage

- Large storage for archival purposes.
- We assume it is not possible to answer queries from the archival store.
- Can be used only under special circumstances using time-consuming retrieval processes.

## Working Store

- Holds summaries or parts of streams.
- Can be used for answering queries.
- Might be in disk or in main memory.
- Cannot store all the data from all the streams.



# Types of Queries

## Standing Queries

- Queries that are always performed on the data.
- In a sense, these are queries that are permanently executing.
- Since these queries are known in advance, it is fairly easy to design efficient storage and query processes to handle them.

## Ad-Hoc Queries

- Queries that are not known in advance.
- These queries are created, for example, by a user or operator.
- We need to find a way to query the current state of the stream.

# Types of Queries

## Standing Queries

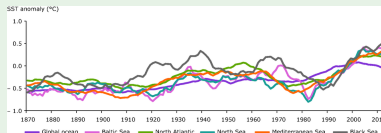
- Queries that are always performed on the data.
- In a sense, these are queries that are permanently executing.
- Since these queries are known in advance, it is fairly easy to design efficient storage and query processes to handle them.

## Ad-Hoc Queries

- Queries that are not known in advance.
- These queries are created, for example, by a user or operator.
- We need to find a way to query the current state of the stream.

# Examples of Standing Queries

## Example: Ocean Surface Temperature Sensor



- 1 Alert when the temperature exceeds 25 degrees centigrade.
- 2 Average the 24 most recent readings.
- 3 Maximum temperature ever recorded.
- 4 Average temperature.

## Question

What information do we need to keep in the working storage to answer each of these standing queries?

# Examples of Working Storage Needs

## Q1: Alert when the temperature exceeds 25°C

- No information required (we do not need to keep any samples in the working storage).

## Q2: Average the 24 most recent readings

- 24 variables, one per reading.

## Q3: Maximum temperature ever recorded

- 1 variable with the value of the maximum so far.

## Q4: Average temperature of all readings so far

- 1 variable with the value of the sum of readings so far.
- 1 variable that counts the number of readings so far.

## Examples of Working Storage Needs

### Q1: Alert when the temperature exceeds 25°C

- No information required (we do not need to keep any samples in the working storage).

### Q2: Average the 24 most recent readings

- 24 variables, one per reading.

### Q3: Maximum temperature ever recorded

- 1 variable with the value of the maximum so far.

### Q4: Average temperature of all readings so far

- 1 variable with the value of the sum of readings so far.
- 1 variable that counts the number of readings so far.

# Examples of Working Storage Needs

## Q1: Alert when the temperature exceeds 25°C

- No information required (we do not need to keep any samples in the working storage).

## Q2: Average the 24 most recent readings

- 24 variables, one per reading.

## Q3: Maximum temperature ever recorded

- 1 variable with the value of the maximum so far.

## Q4: Average temperature of all readings so far

- 1 variable with the value of the sum of readings so far.
- 1 variable that counts the number of readings so far.

# Examples of Working Storage Needs

## Q1: Alert when the temperature exceeds 25°C

- No information required (we do not need to keep any samples in the working storage).

## Q2: Average the 24 most recent readings

- 24 variables, one per reading.

## Q3: Maximum temperature ever recorded

- 1 variable with the value of the maximum so far.

## Q4: Average temperature of all readings so far

- 1 variable with the value of the sum of readings so far.
- 1 variable that counts the number of readings so far.

## Question: An effective way to compute the average temperature

### Q4: Average temperature

If we keep the sum of readings so far we may have problems with data overflow (the sum may exceed the capacity of storage)

- 1 How serious is this problem?
- 2 How could we fix this problem?



# Examples of Ad-hoc Queries

## Example: Web Site

- 1 What were the unique users in the past month?
- 2 What were the users from Australia?
- 3 What were the users with generated most traffic?

## Note

- If the above were questions were known beforehand they would be standing queries.
- Given an application we can optimise it to enable the processing of some kinds of ad-hoc queries.
- In general, it is impossible to be able to accurately answer all possible ad-hoc queries.

# Programme

- 1 Data Streams
- 2 The Stream Model
- 3 Technologies for Stream Analytics
  - StreamSQL
  - Machine Learning on Streams

# Some Platforms for Stream Analytics

- Azure Stream Analytics  
<https://azure.microsoft.com/en-au/services/stream-analytics/>
- Amazon Kinesis <https://aws.amazon.com/kinesis/>
- Apache Flink <https://flink.apache.org/>
- Apache Kafka <https://kafka.apache.org/>
- SAS Event Stream Processing  
<https://www.sas.com/en-au/software/event-stream-processing.html>
- SQLStream <https://sqlstream.com/>
- IBM Streaming Analytics  
<https://www.ibm.com/cloud/streaming-analytics>

# Programme

- 1 Data Streams
- 2 The Stream Model
- 3 Technologies for Stream Analytics
  - StreamSQL
  - Machine Learning on Streams

# StreamSQL

- StreamSQL is a query language that extends SQL with the ability to process real-time data streams.
- Various platforms for stream analytics incorporate their own versions of StreamSQL.
  - Apache Flink uses Apache Calcite's proposal  
<https://calcite.apache.org/docs/stream.html>.
  - Apache Kafka uses Confluent KSQL  
<https://www.confluent.io/product/ksql/>.
  - Azure Stream Analytics uses a subset of Transact-SQL  
<https://msdn.microsoft.com/en-us/azure/stream-analytics/reference/stream-analytics-query-language-reference>.
- Can be linked to [event stream processing](#).
  - The StreamSQL query defines a pattern to be captured in an event.

# StreamSQL Example 1

## Example

This example defines a standing SQL query that is continuously triggered and processes the last second of a stream.

[https://en.wikipedia.org/wiki/Event\\_stream\\_processing](https://en.wikipedia.org/wiki/Event_stream_processing)

```
SELECT DataStream
    Orders.TimeStamp, Orders.orderId , Orders.ticker ,
    Orders.amount, Trade.amount
FROM Orders
JOIN Trades OVER (RANGE INTERVAL '1' SECOND FOLLOWING)
ON Orders.orderId = Trades.orderId;
```

## StreamSQL Example 2

### Example

This example defines a standing SQL query that is triggered when a man wearing tuxedo appears, followed by a person wearing a gown and either church bells or flying rice.

[https://en.wikipedia.org/wiki/Event\\_stream\\_processing](https://en.wikipedia.org/wiki/Event_stream_processing)

```
WHEN Person.Gender EQUALS "man" AND  
      Person.Clothes EQUALS "tuxedo"  
FOLLOWED-BY  
      Person.Clothes EQUALS "gown" AND  
      (Church_Bell OR Rice_Flying)  
WITHIN 2 hours  
ACTION Wedding
```

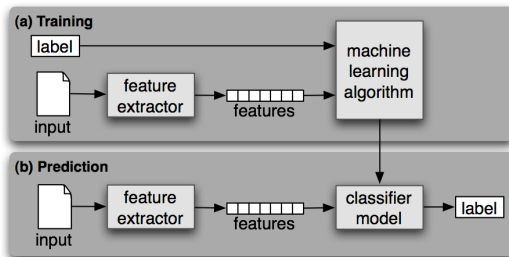
# Programme

- 1 Data Streams
- 2 The Stream Model
- 3 Technologies for Stream Analytics
  - StreamSQL
  - Machine Learning on Streams



# Machine Learning on Streams

- Supervised approaches for machine learning require training data.
- It is important that the training data must be a representative sample of the real data.
- But data in streams never ends.
- Even worse, data in streams may change in time.



## Solution 1: Training with Batches

- Re-train the system regularly.
  - The frequency of re-train depends on how fast the data changes.
- If a lot of data has been generated since last training, keep a sample of the training data.
  - E.g. keep the most recent data from the stream for training.

### BUT

- The system may not handle unexpected drifts in the data.
- Re-training can take much computation time and resources.

## Solution 2: On-line Machine Learning

- Keep an infinite training loop.
- Update the trained model from data sampled from the stream.

Initialise model parameters;

**while** *True* **do**

    Sample from the stream;

    Update model parameters;

    Save model for production;

**end**

# Take-home Messages

- Applications of Stream Processing.
- The Four V's of Big Data for Stream Processing.
- The Stream Model.
- StreamSQL.
- Machine Learning on Streams.

# What's Next

## Week 12

- Invited Lecture (TBA).