Eötvös Loránd University (ELTE)
Faculty of Informatics (IK)
Pázmány Péter sétány 1/c
1117 Budapest, Hungary

STREAMING BASICS

Stream mining (SM)

Imre Lendák, PhD, Associate Professor
Péter Kiss, PhD candidate

2020
Budapest, Hungary

Outline

- Streaming basics
- Reasoning about time
- Batch data processing 101
- Stream processing 101
- Triggers
- Watermarks
- Accumulation

Stream mining

STREAMING BASICS

Why streaming?

- Businesses need timely (i.e. immediate) insights into their data
- Massive, unbounded datasets are increasingly common in different business domains
- Processing data as it arrives spreads workloads more evenly over time → consistent and predictable consumption of computing resources (e.g. if we rent cloud-based resources)
  - This is the opposite of hoarding large amounts of data and periodically running high CPU/memory use analyses

Kinds of streams

- Click streams
- Sensor measurements
- Satellite imaging data
- Power grid electricity distribution
- Banking/e-commerce transactions
- Security monitoring data

Necessary definitions

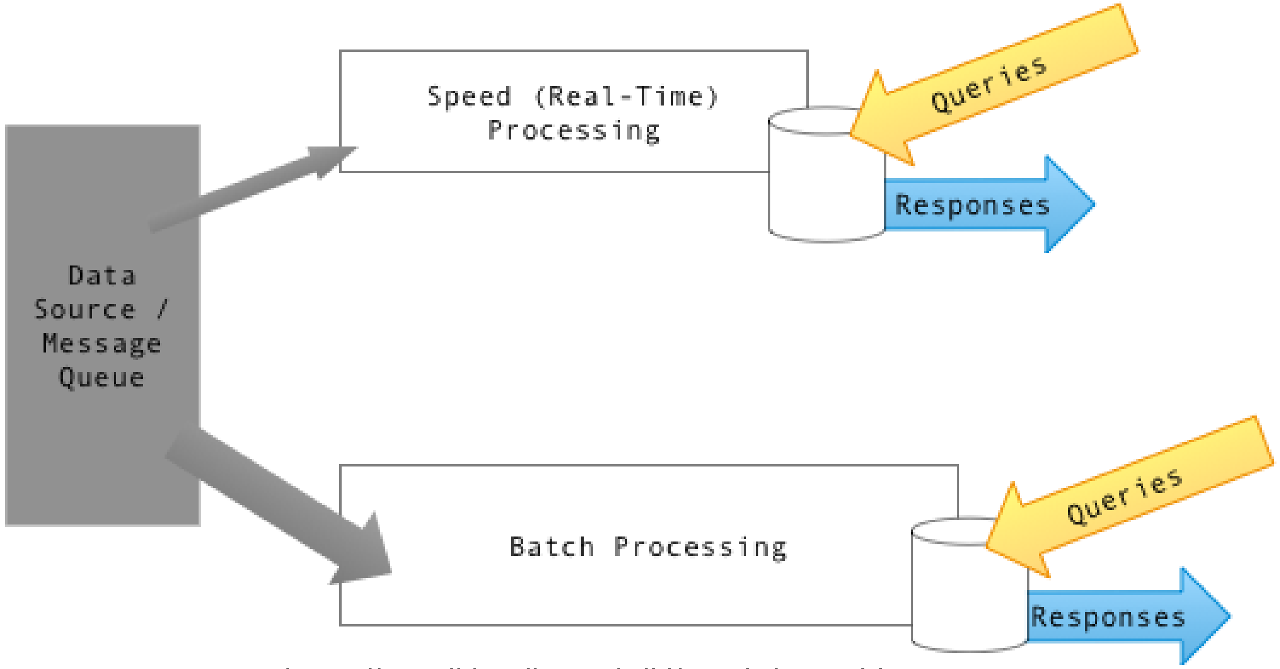
- Dataset cardinality
  - DEF: **Bounded data** is finite in size.
  - DEF: **Unbounded data** is infinite in size.
- Data constitution
  - DEF: A **table** represents a holistic (~complete) view of a dataset at a specific point in time.
  - DEF: A **stream** is an element-by-element view of the evolution of a dataset over time.
    - Alternate definition: A data stream is an ordered (not necessarily always) and potentially infinite sequence of data points (e.g. numbers, words, sequences).
- DEF: A **streaming system** is a data processing engine designed for handling infinite (unbounded) datasets.

Traditional streaming

- Characteristics of traditional streaming systems:
  - The good: low latency
  - The bad: inaccurate, i.e. lack of consistency → non-deterministic

- **DEF:** Batch systems are deterministic as they provide eventually correct results, i.e. once all relevant data is acquired and analyzed

Lambda architecture



https://en.wikipedia.org/wiki/Lambda_architecture

- Lambda architecture: running a traditional streaming system **in parallel** with a batch data analysis solution
  - The good: low-latency (though inaccurate) results from the streaming element, correct results from the batch subsystem
  - The bad: hassle to implement and maintain (2 systems!)

‘Modern’ streaming

1. Correctness
   - Consistent storage
   - Exactly-once processing
2. Reasoning about time
   - Techniques for reasoning about time in the presence of unbounded, unordered data of varying event time skew

Stream mining

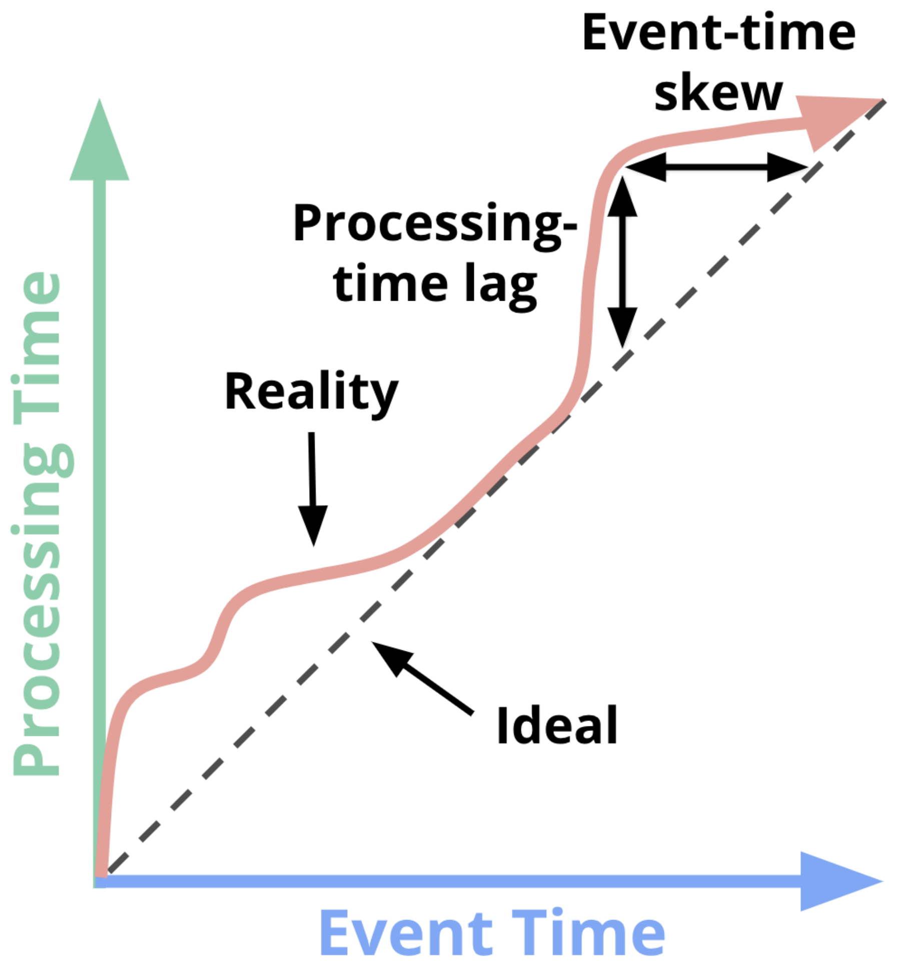
REASONING ABOUT TIME

Processing vs event time

- **Processing time** = The time at which events are observed by the system
- **Event time** = The time at which events actually occurred
- In non-ideal systems there can be a lag between processing and event times caused by a temporary lack of network connectivity, e.g. mobile device in airplane mode, maintenance crew doing repairs of power lines in remote (geographic) locations

- In industrial control systems and general purpose supervisory control and data acquisition systems (SCADA) event time is sometimes called ‘server’ time and processing time is ‘client’ time

Event time vs processing time



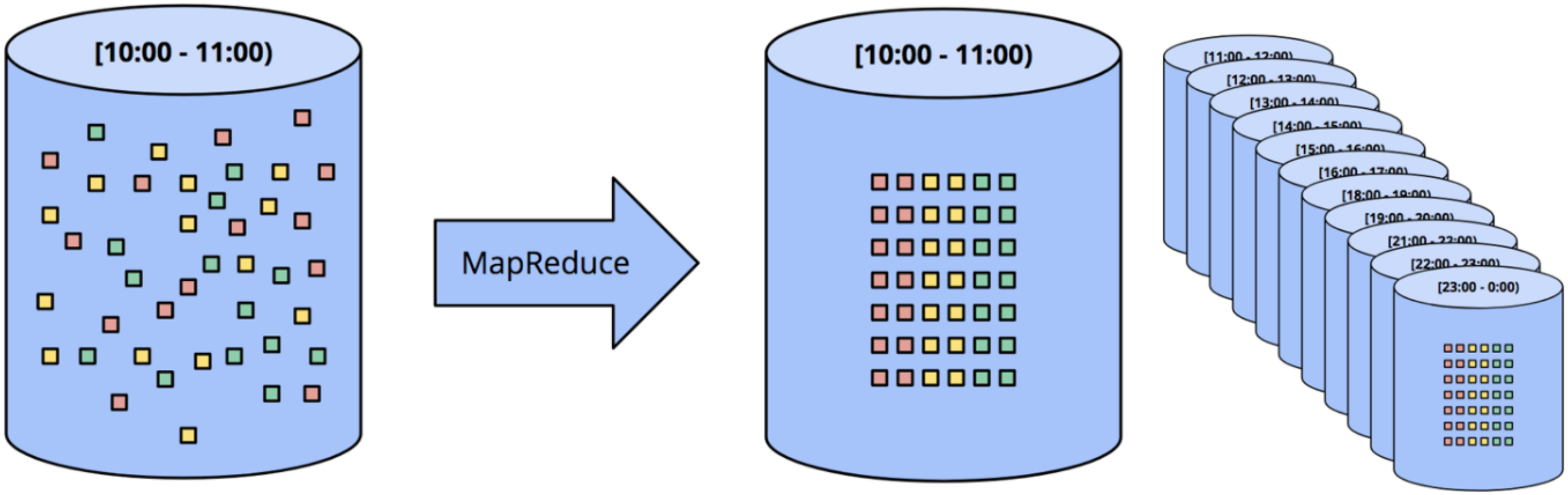
http://streamingsystems.net/fig/1-1

- X axis → event-time completeness in the system → the time X in event time up to which all data with event times less than X have been observed.
- Y axis → the progress of processing time → normal clock time as observed by the data processing system as it executes.

Stream mining

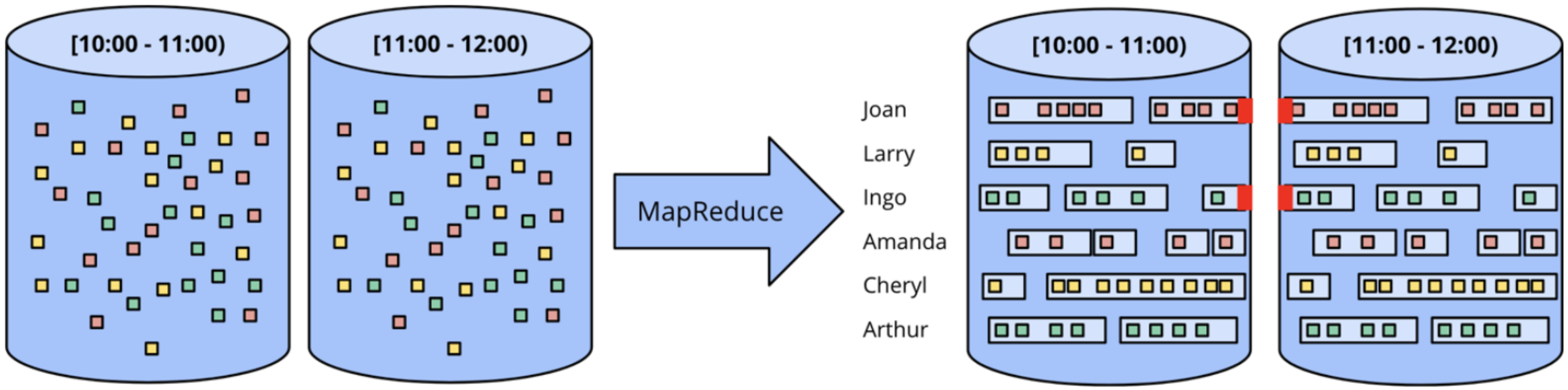
BATCH DATA PROCESSING 101

Fixed windows (in processing time)



http://streamingsystems.net/fig/1-3

Sessions


[10:00 - 11:00)
[11:00 - 12:00)
MapReduce
Joan
Larry
Ingo
Amanda
Cheryl
Arthur
[10:00 - 11:00)
[11:00 - 12:00)

Stream mining

STREAM PROCESSING 101

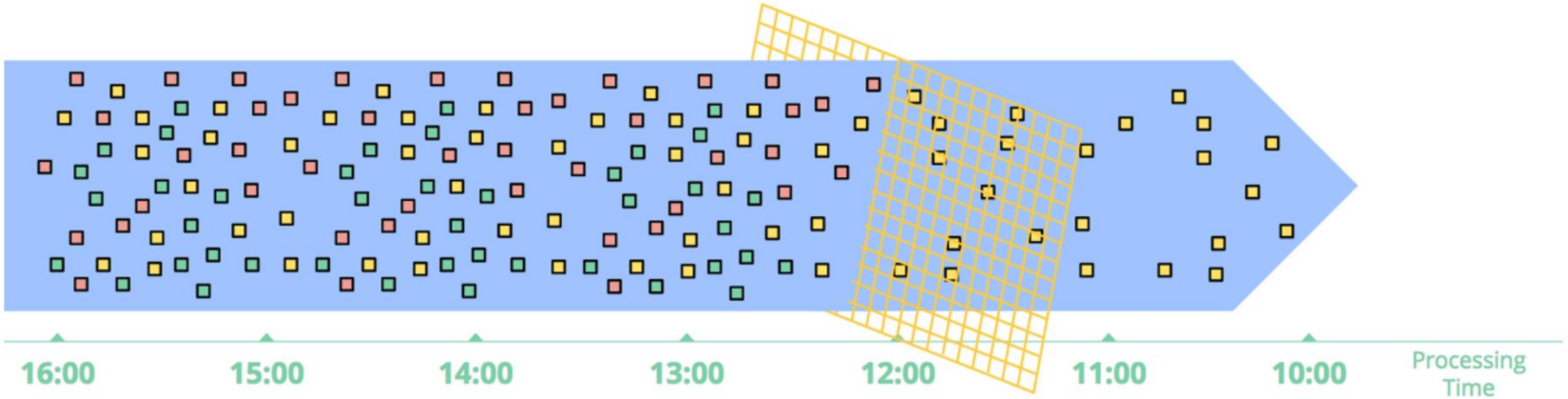
Stream data characteristics

- Unbounded → potentially infinite incoming data
- Highly unordered in respect to event time → it is necessary to ‘shuffle’ data in the processing pipeline
- Varying event time skew → we cannot assume that most data with event time X will arrive within constant epsilon time Y

Potential processing solutions

- Depending on the specific requirements of the data processor, we differentiate the following approaches to stream processing
  - Time-agnostic
  - Approximation
  - Windowing by processing time
  - Windowing by event time

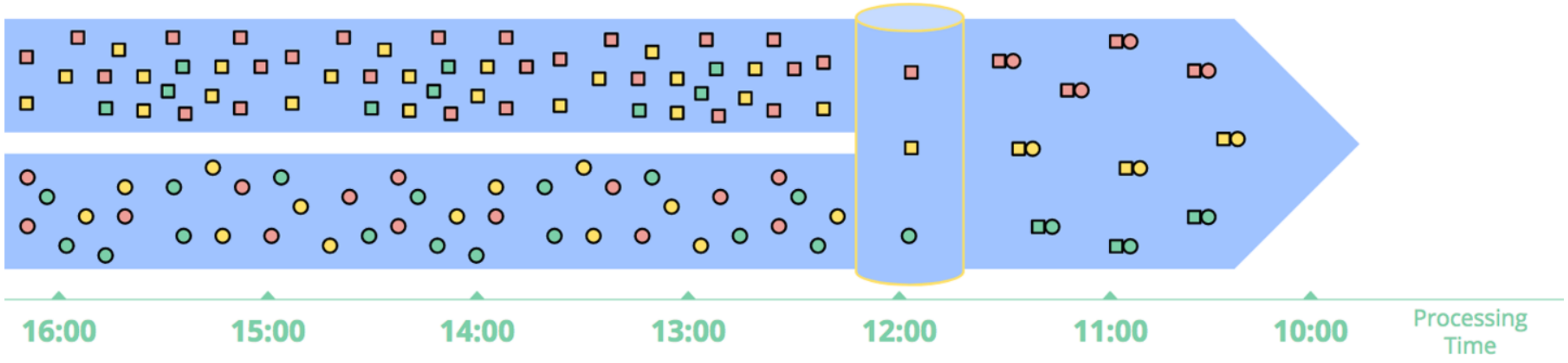
Time agnostic #1: Filtering



http://streamingsystems.net/fig/1-5

- Look for data which satisfy a certain filter, e.g. social network users from China & disregard all else

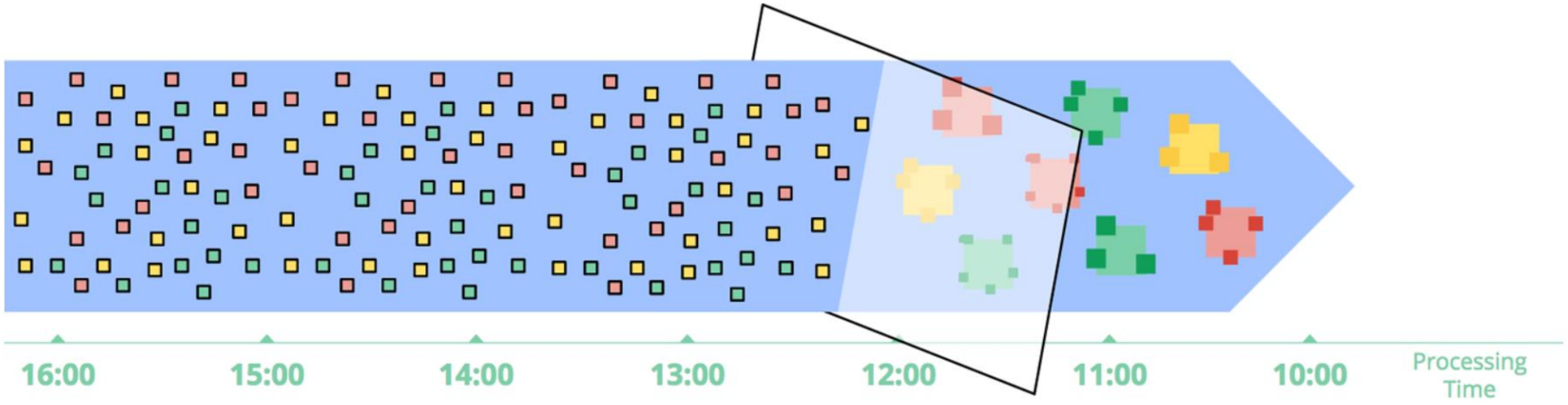
Time agnostic #2: Inner joins



http://streamingsystems.net/fig/1-6

- Wait until (a) certain piece(s) of data (marked in red, yellow and green in the figure above) appear(s) in multiple input (data) sources, join them and continue

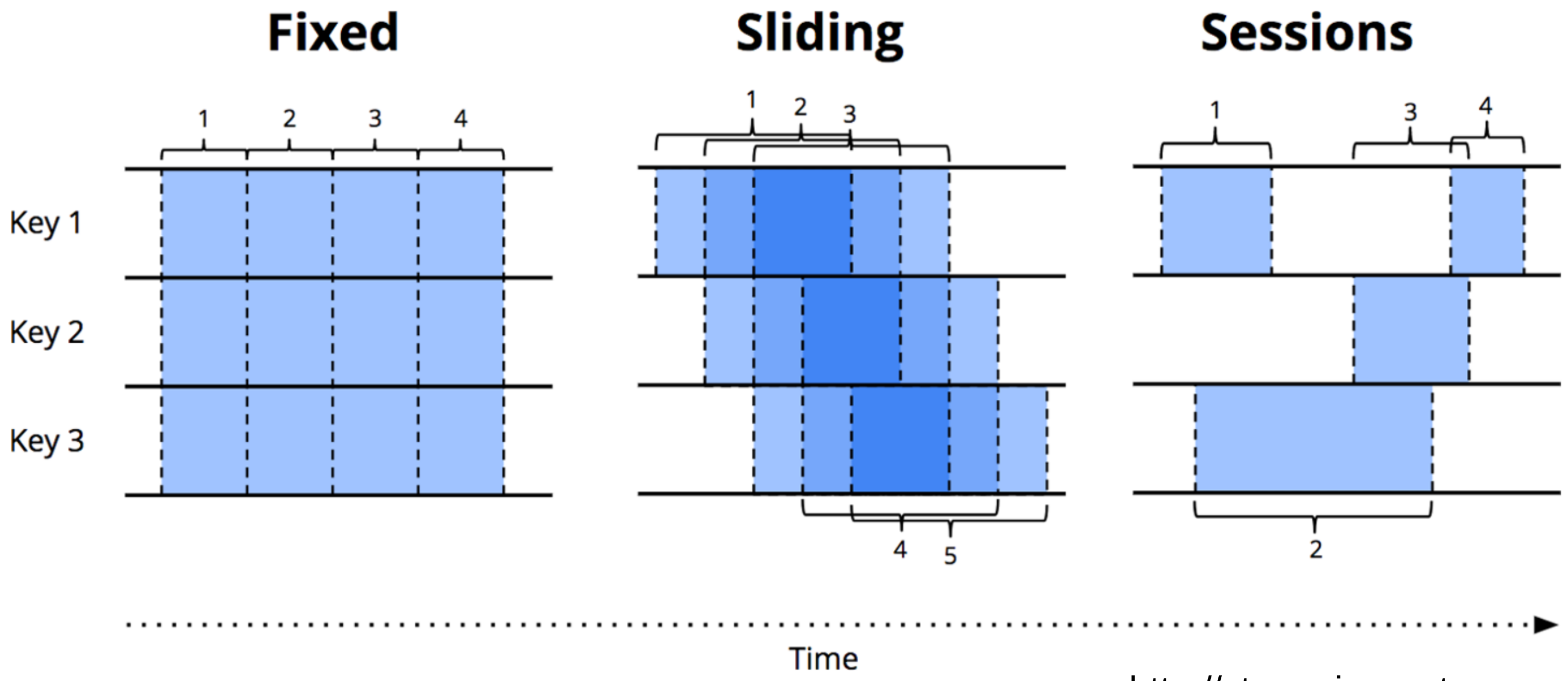
Approximations



http://streamingsystems.net/fig/1-7

- Goal: ‘look’ at the data and make educated guesses, i.e. approximations
- Pro: low overhead
- Contra: limited number of such algorithms, challenging topic with limited numbers of new contributionsS

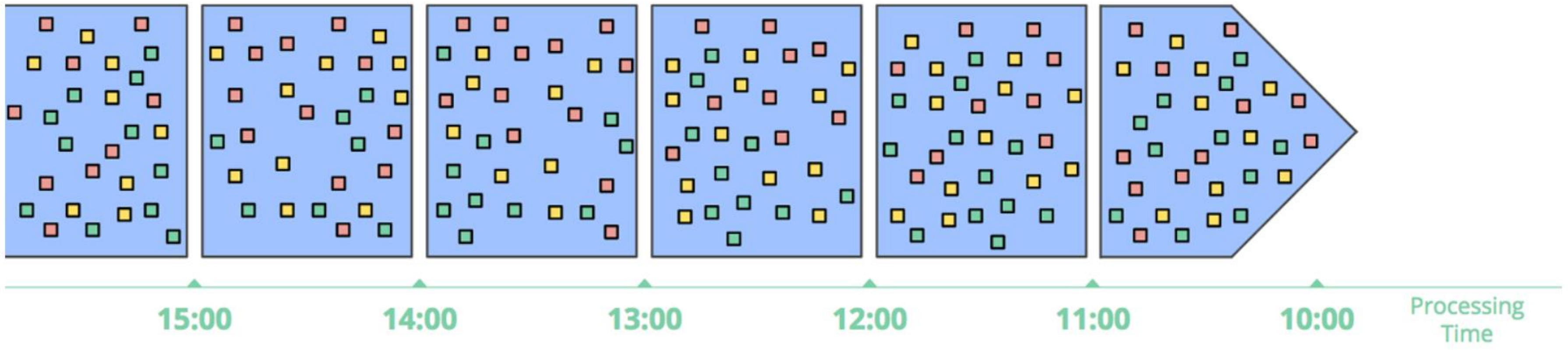
General windowing strategies



http://streamingsystems.net/fig/1-8

- Fixed windows → Slice time into fixed-sized temporal length, usually across all keys/variables.
- Sliding windows → Defined by fixed length & fixed period.
- Sessions → Sequences of events terminated by a gap.

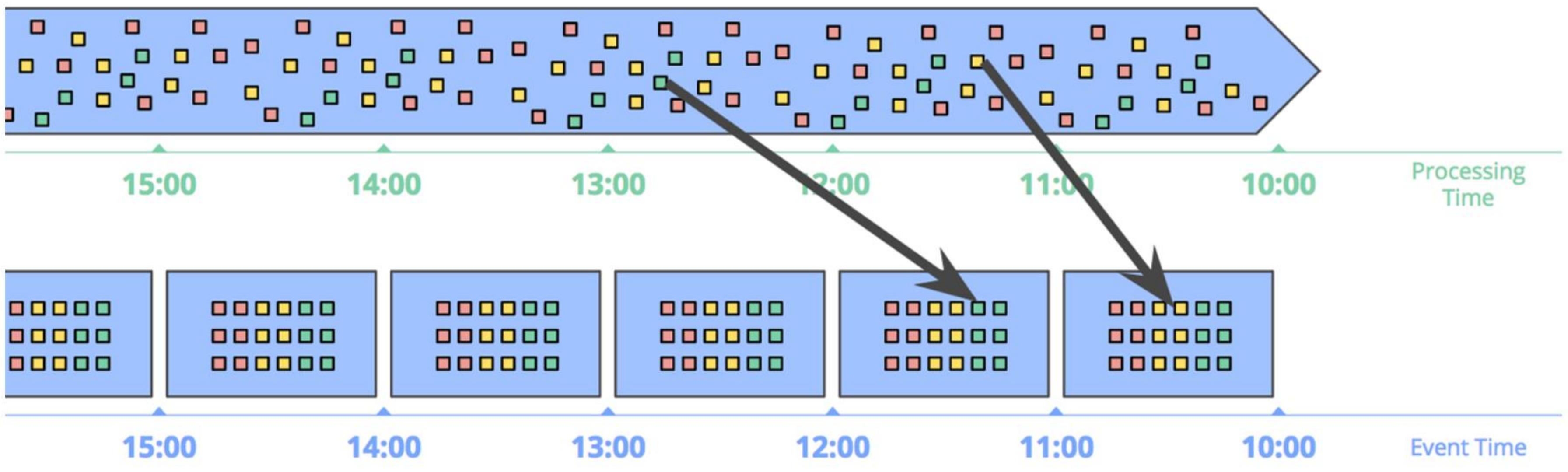
Windowing by processing time



http://streamingsystems.net/fig/1-9

- Windows are created based on absolute (processing) time, i.e. data is put into windows based on the order they arrive in

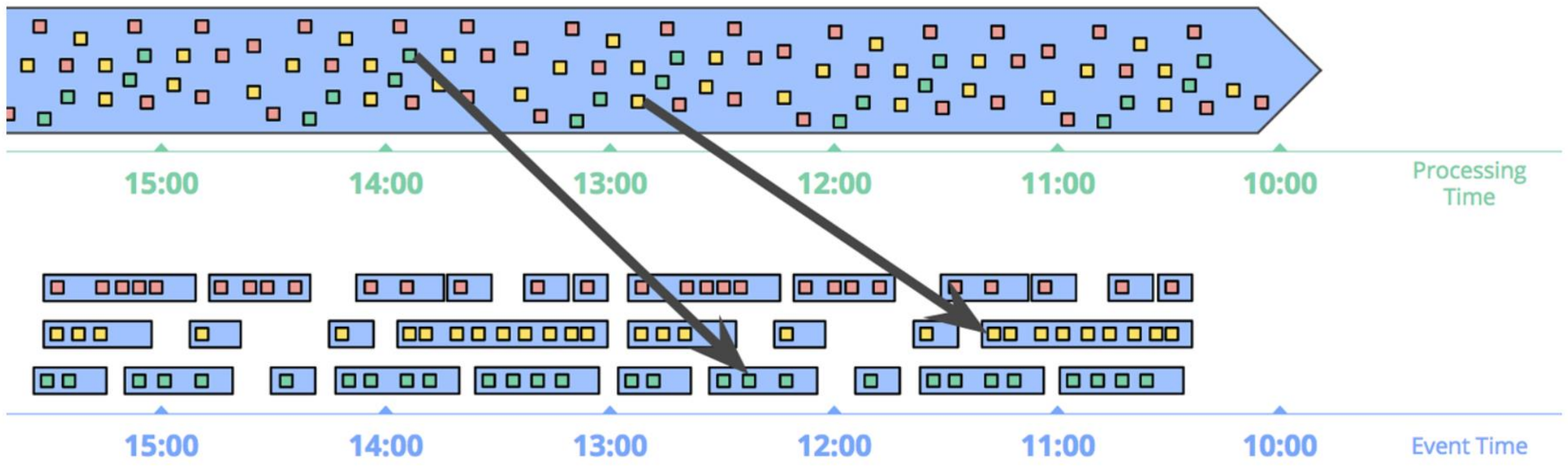
Windowing by event time – Fixed



http://streamingsystems.net/fig/1-10

- Data is collected and windowed based on times at which it occurred.

Windowing by event time – Sessions



http://streamingsystems.net/fig/1-11

- Data is collected into sessions based on temporal proximity and key/user.

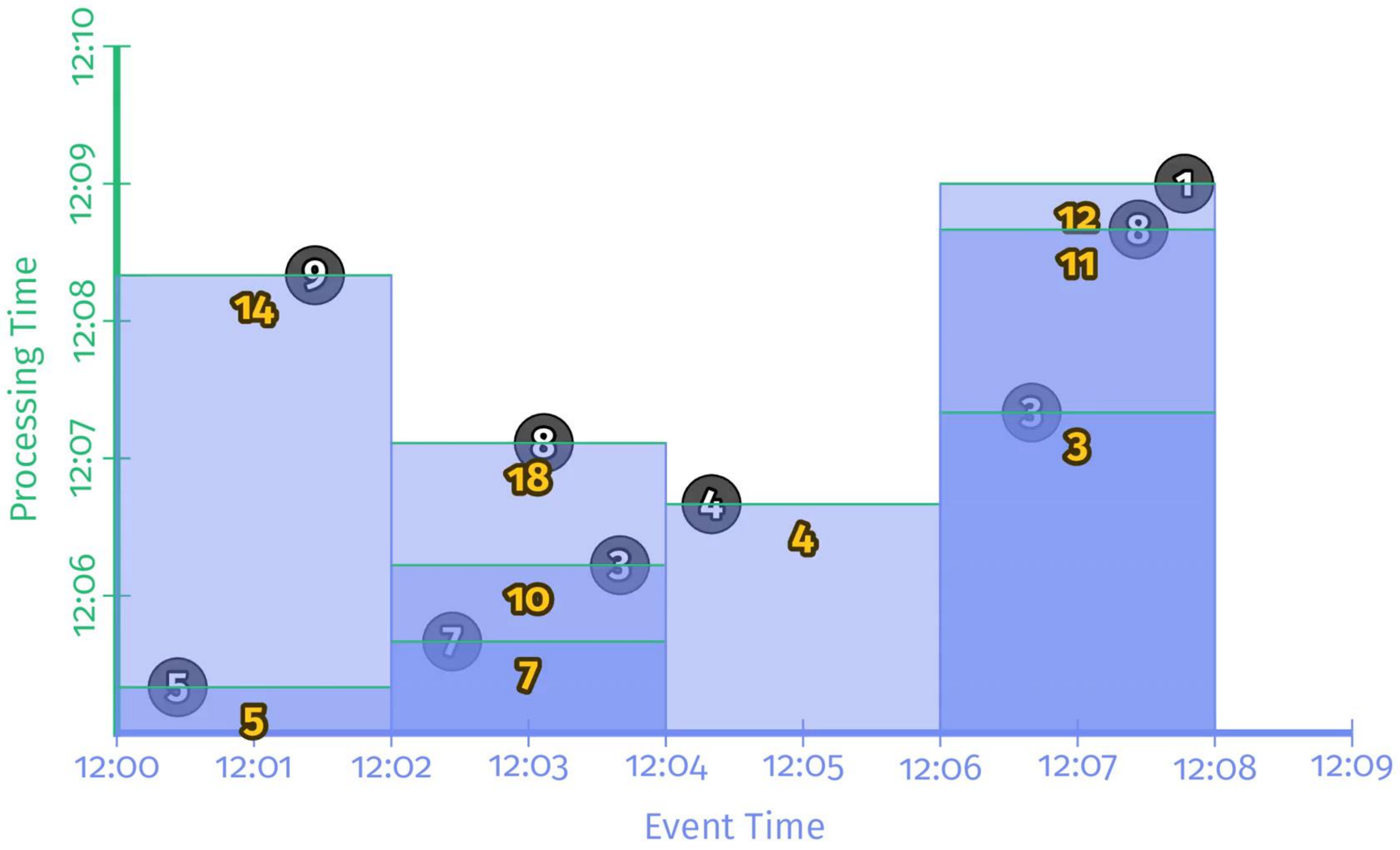
Stream mining

TRIGGERS

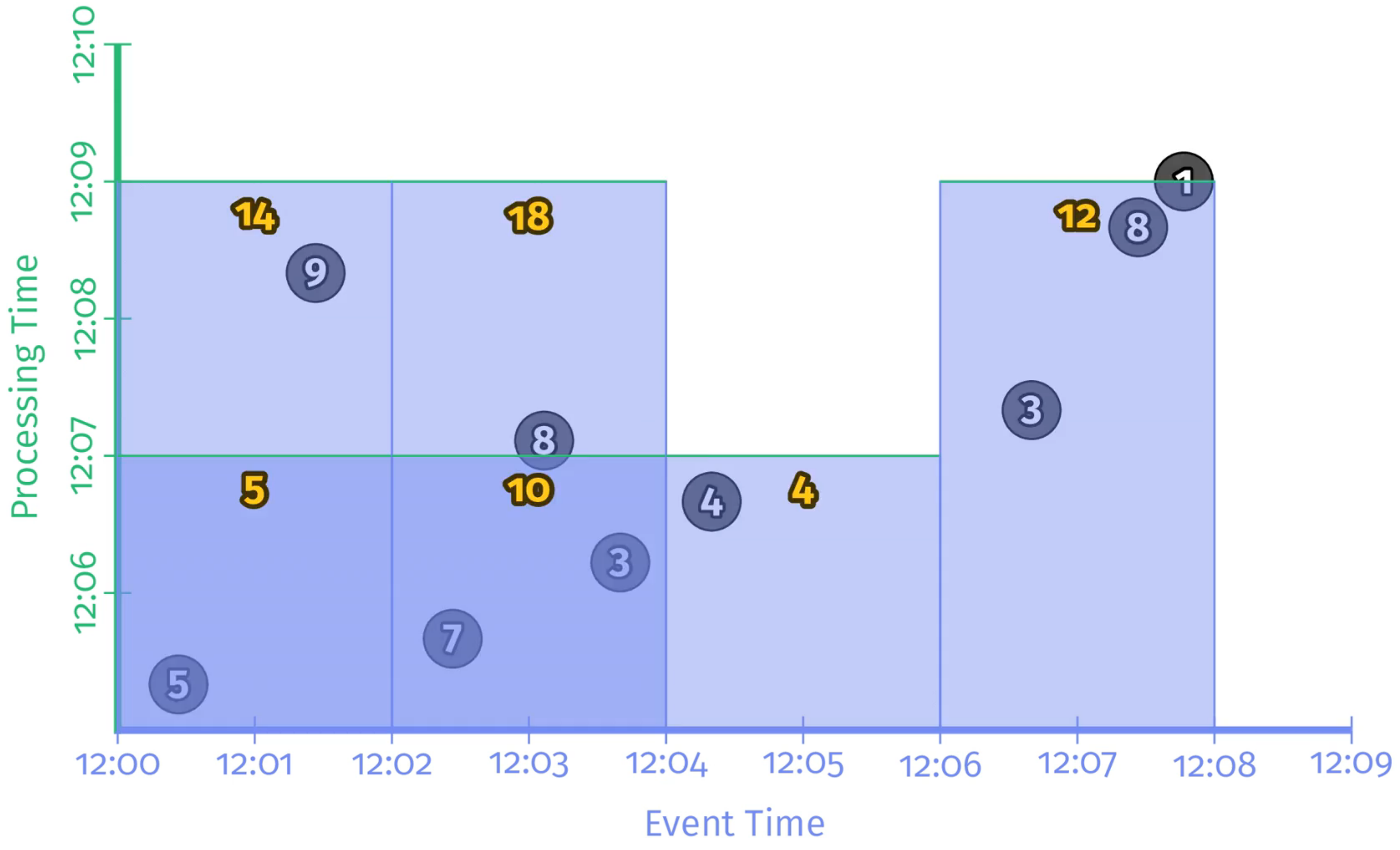
Triggers & variants

- **DEF:** Triggers declare when output for a window should happen in processing time, i.e. when data should be processed
- Trigger types:
  - **Repeated update triggers** = most common in streaming systems, generate periodic updates for a window. Updates materialized for
    - Every new record → too many processing → high CPU load
    - After some processing time delay, e.g. 5 minutes
  - **Completeness triggers** = materialize each time a window is (believed to be) complete to certain degree, e.g. ~80% of data, ~90% of data, etc.
    - Allow reasoning about missing and late data

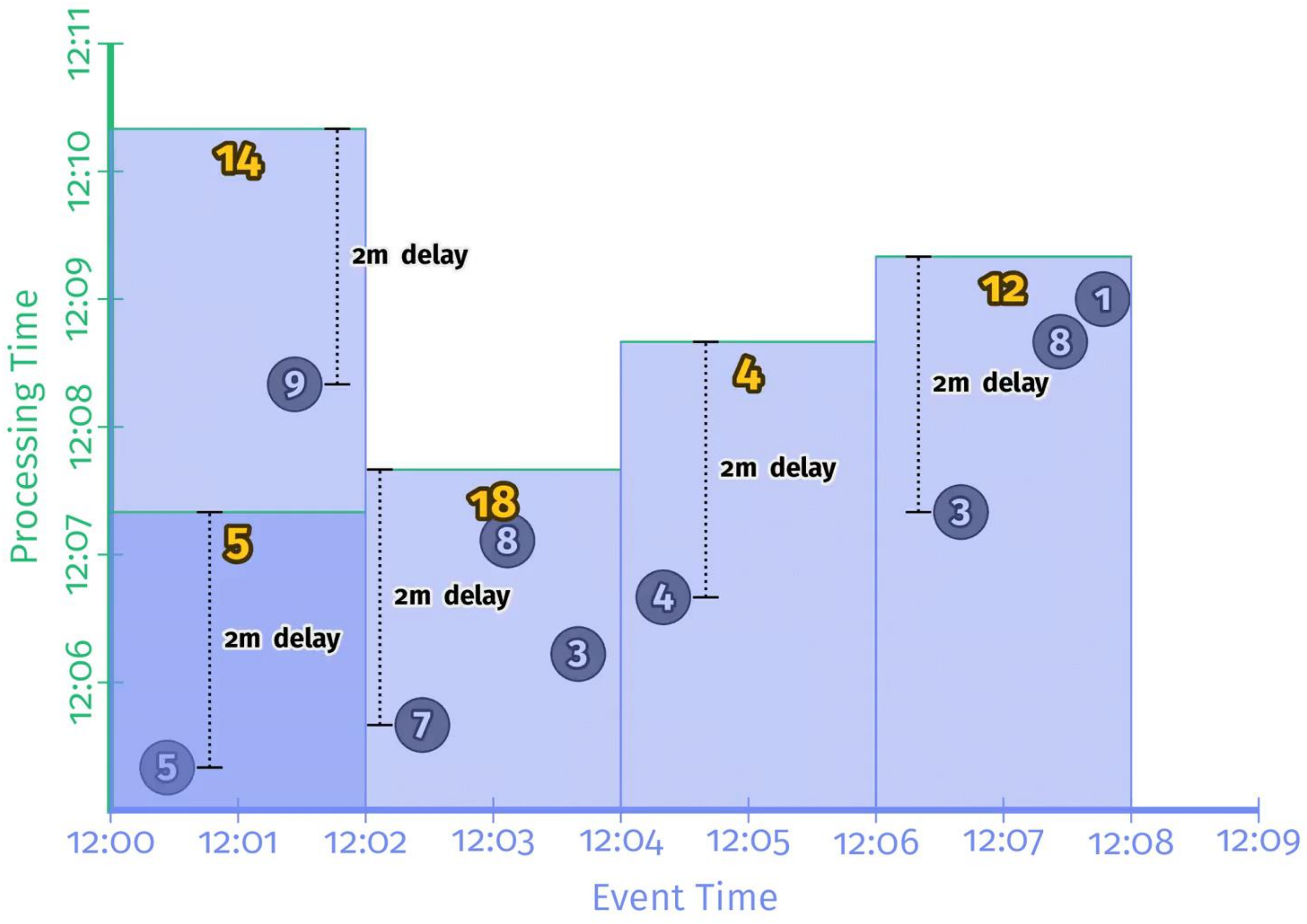
Per record triggering



2-minute aligned delay triggering ~ microbatch streaming



2-minute unaligned delay triggering



Stream mining

WATERMARKS

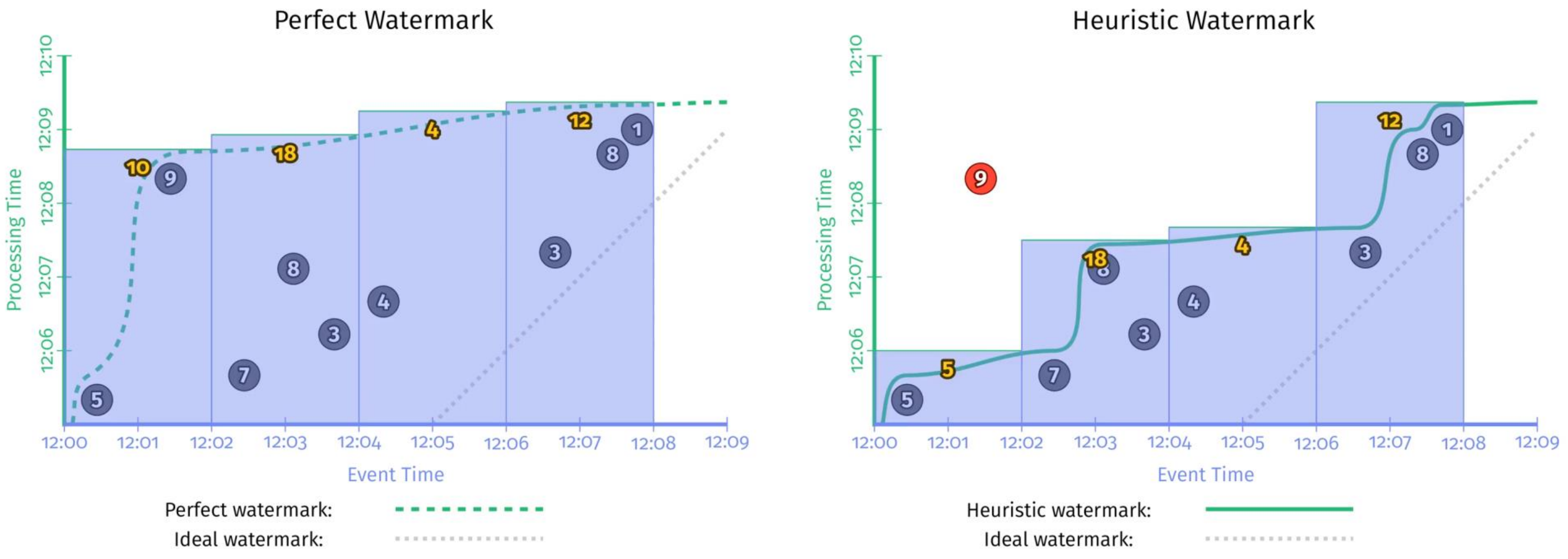
Watermarks

- **DEF:** Watermarks are temporal notions of input completeness in the event time domain
  - ‘Guesses’ about data completeness, i.e. all relevant data received
- Alternative definition: Watermarks allow streaming systems to measure progress and completeness relative to event times of the records being processed

Watermark types

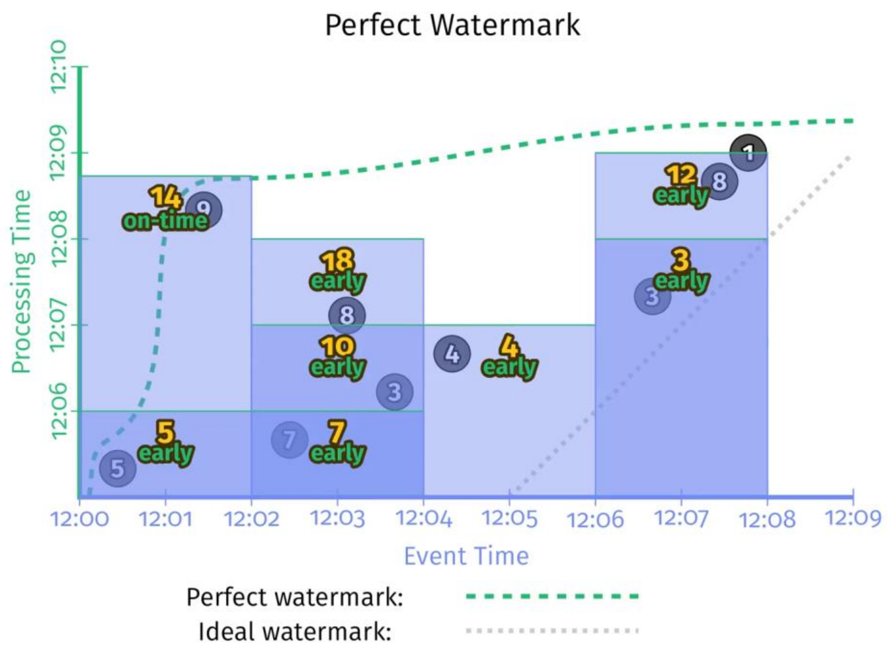
- **Perfect watermark** = usable when we have perfect knowledge about the input data → all data on time (i.e. no late data
- **Heuristic watermark** = provide an estimate of (data) progress as good as it gets
  - Problem #1: ‘Too slow’ = the watermark is correctly delayed due to known unprocessed data, e.g. due to network delays
  - Problem #2: ‘Too fast’ = when a watermark is advanced earlier than it should be → relevant data is discarded and not processed

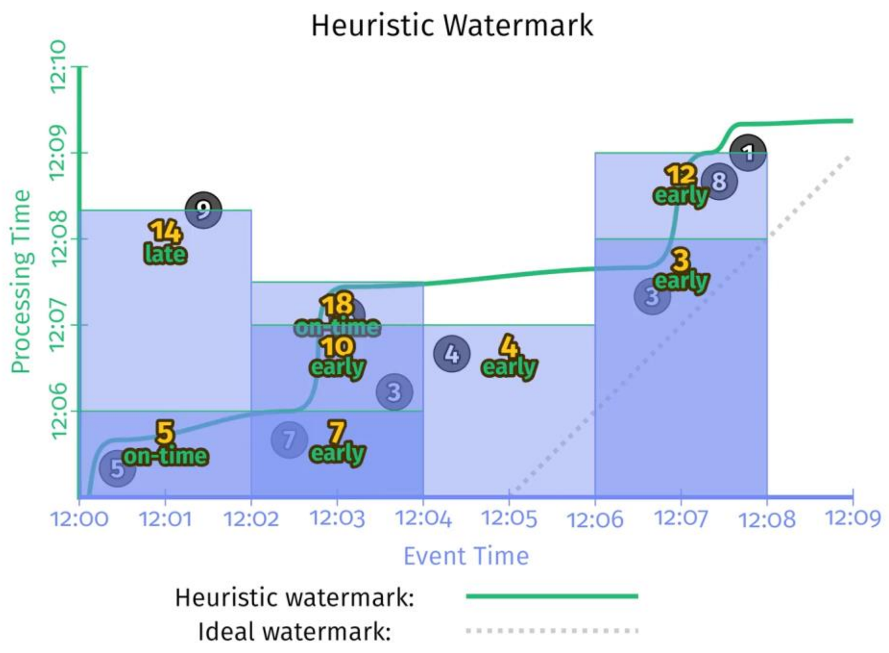
Perfect vs heuristic watermarks



http://streamingsystems.net/fig/2-10

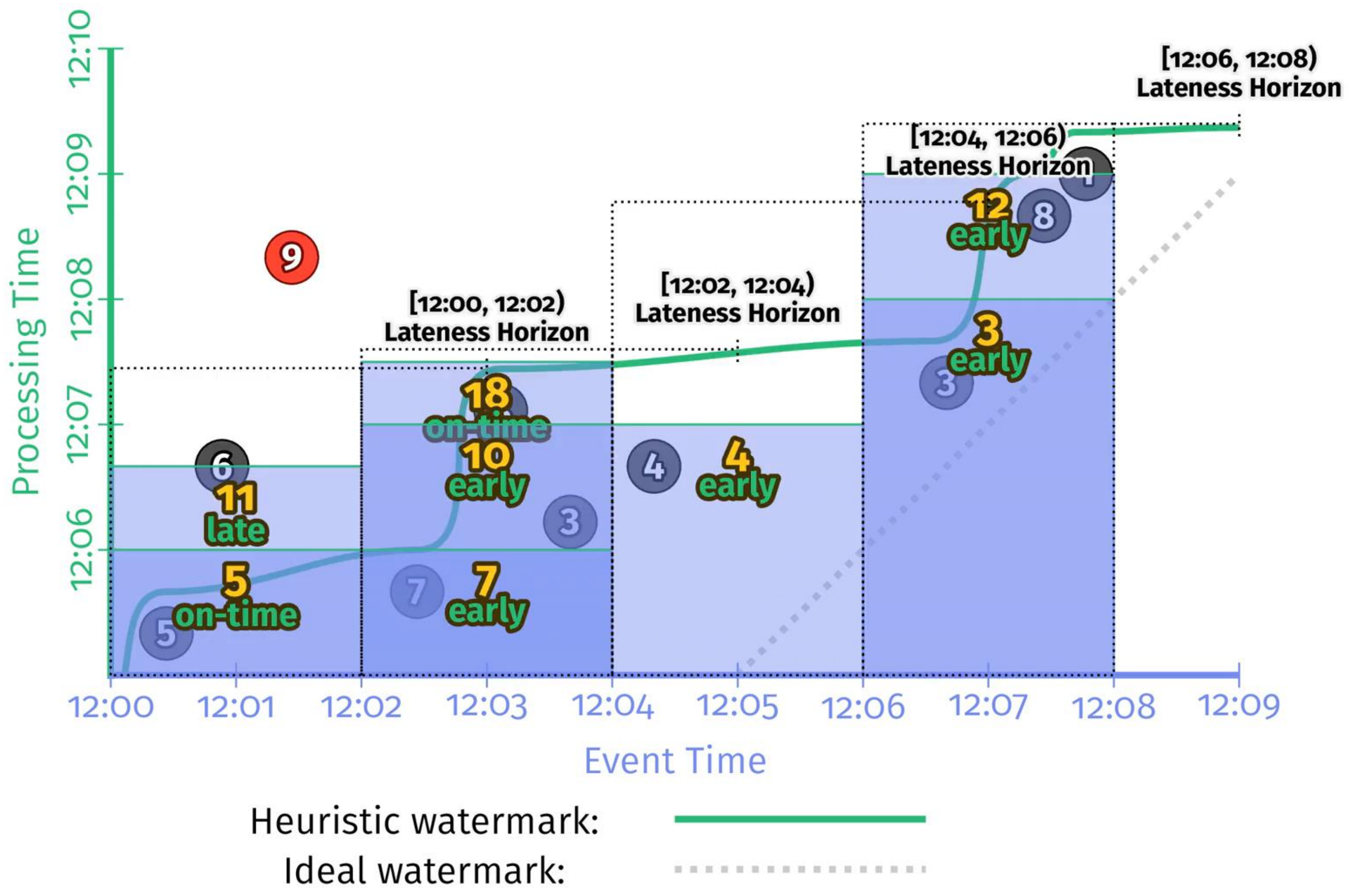
Watermarks meet triggers





- Early/on-time/late triggers
  - Zero or more early triggers **periodically** firing
  - Single **on-time** trigger on completeness/watermark
  - Zero or more triggers for **late data** unaccounted for by the watermark

Allowed lateness



http://streamingsystems.net/fig/2-12

Stream mining

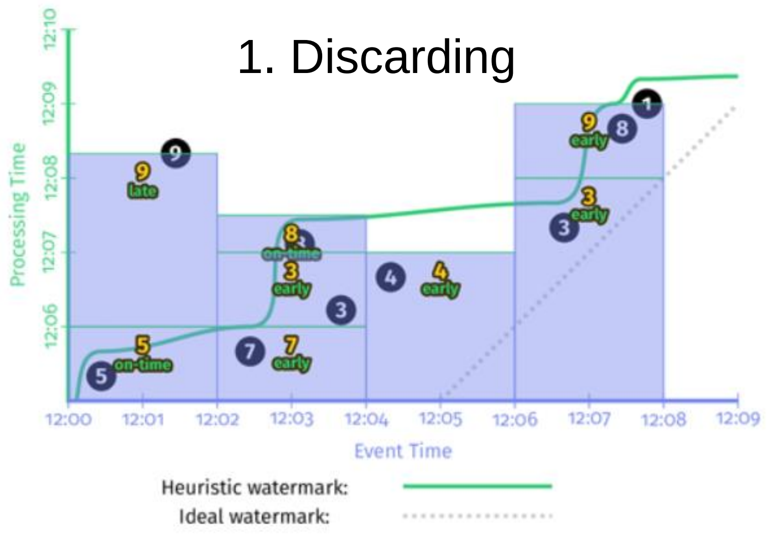
ACCUMULATION

Accumulation intro

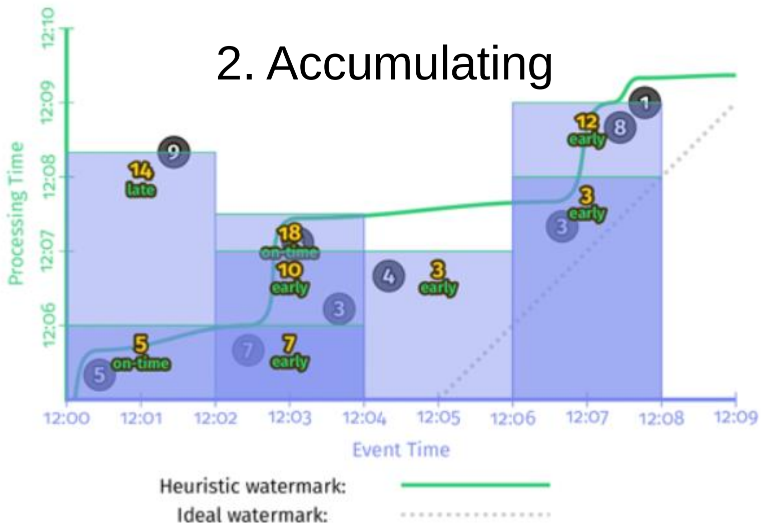
- Types of accumulation when triggers produce multiple results (processing or panes) during a single window
  - **Discarding** = any prior data originating from an earlier processing triggered during the same window is **discarded**
  - **Accumulating** = stored state is retained (as opposed to discarding) – useful for generating aggregates based on ‘deltas’
  - **Accumulating and retracting** = an extension of the accumulating approach, but additionally retaining the difference between the current processed value and the prior value, e.g. if the current sum is 12 and the previous sum was 9, then retain (12, -3)

Accumulation types side-by-side

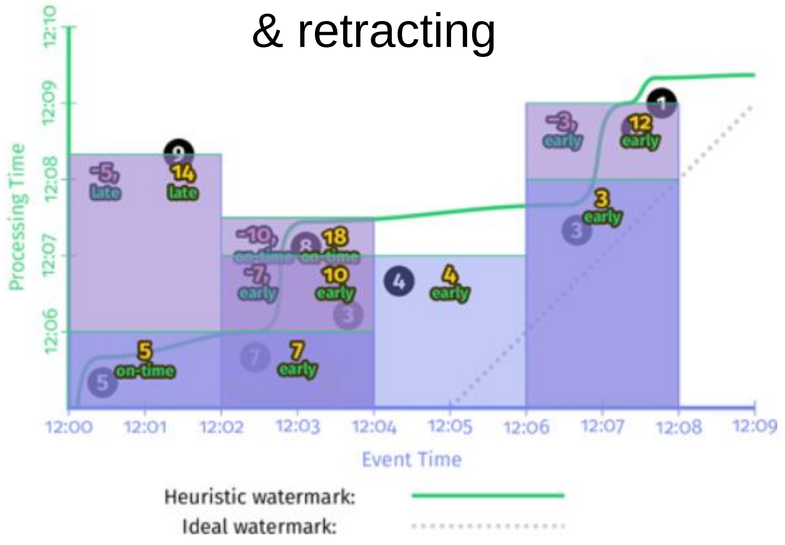
1. Discarding



2. Accumulating



3. Accumulating & retracting



Summary

- Streaming basics
- Reasoning about time
- Batch data processing 101
- Stream processing 101
- Triggers
- Watermarks
- Accumulation

Thank you for your attention!