CS 744: DATAFLOW

Shivaram Venkataraman

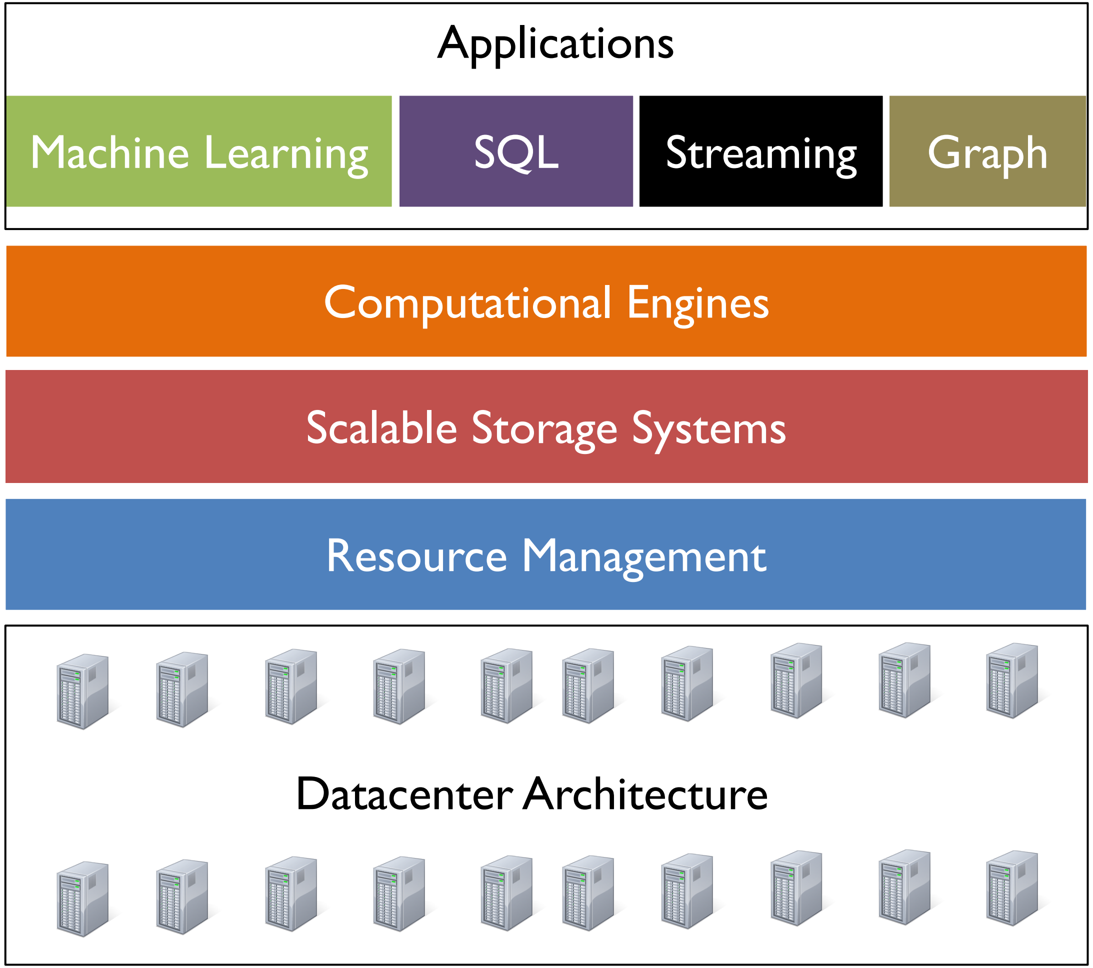
Fall 2019

Administrivia

- Assignment 2 grades up
- Midterm grading
- Course project proposal comments

- AEFIS feedback

- No Class next Tuesday?


Applications
Machine Learning
SQL
Streaming
Graph
Computational Engines
Scalable Storage Systems
Resource Management
Datacenter Architecture

DATAFLOW MODEL (?)

MOTIVATION

Streaming Video Provider

- How much to bill each advertiser ?
- Need per-user, per-video viewing sessions
- Handle out of order data

Goals

- Easy to program
- Balance correctness, latency and cost

APPROACH

API Design

- Separate user-facing model from execution
- Decompose queries into
  - What is being computed
  - Where in time is it computed
  - When is it materialized
  - How does it relate to earlier results

TERMINOLOGY

Unbounded/bounded data

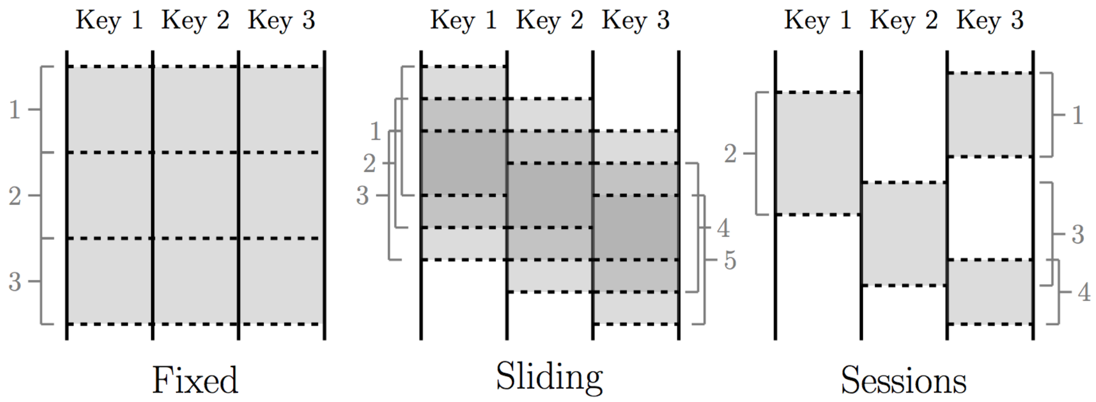
Streaming/Batch execution

Timestamps

Event time:

Processing time:

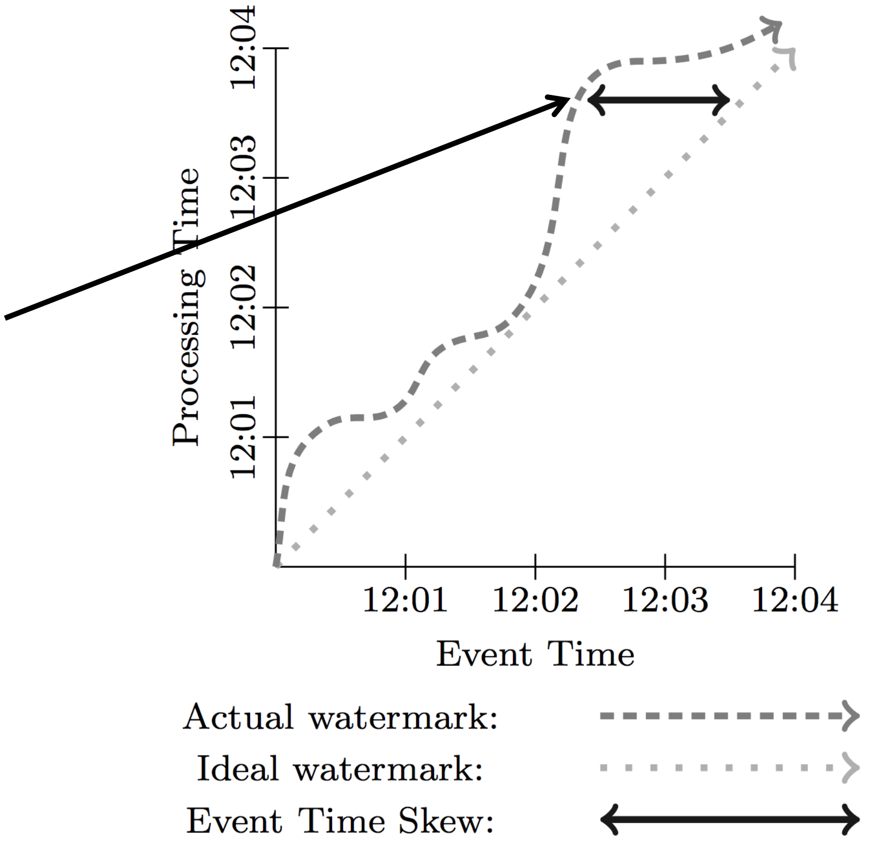
WINDOWING



Key 1 Key 2 Key 3
1
2
3
Fixed
Key 1 Key 2 Key 3
1
2
3
4
5
Sliding
Key 1 Key 2 Key 3
1
2
3
4
Sessions

WATERMARK OR SKEW

System has processed all events up to 12:02:30



API

ParDo:

GroupByKey:

Windowing

AssignWindow

MergeWindow

EXAMPLE

$$\begin{array}{c}(k_1, v_1, 13{:}02, [0, \infty)),\\ (k_2, v_2, 13{:}14, [0, \infty)),\\ (k_1, v_3, 13{:}57, [0, \infty)),\\ (k_1, v_4, 13{:}20, [0, \infty))\end{array}$$

$$\downarrow \; AssignWindows(Sessions(30m))$$

$$\begin{array}{c}(k_1, v_1, 13{:}02, [13{:}02, 13{:}32)),\\ (k_2, v_2, 13{:}14, [13{:}14, 13{:}44)),\\ (k_1, v_3, 13{:}57, [13{:}57, 14{:}27)),\\ (k_1, v_4, 13{:}20, [13{:}20, 13{:}50))\end{array}$$

$$\downarrow \; DropTimestamps$$

$$\begin{array}{c}(k_1, v_1, [13{:}02, 13{:}32)),\\ (k_2, v_2, [13{:}14, 13{:}44)),\\ (k_1, v_3, [13{:}57, 14{:}27)),\\ (k_1, v_4, [13{:}20, 13{:}50))\end{array}$$

GroupByKey

$$\begin{array}{c}(k_1, [(v_1, [13{:}02, 13{:}32)),\\ (v_3, [13{:}57, 14{:}27)),\\ (v_4, [13{:}20, 13{:}50))]),\\ (k_2, [(v_2, [13{:}14, 13{:}44))])\end{array}$$

$$\downarrow \; MergeWindows(Sessions(30m))$$

$$\begin{array}{c}(k_1, [(v_1, [\mathbf{13{:}02}, \mathbf{13{:}50})),\\ (v_3, [13{:}57, 14{:}27)),\\ (v_4, [\mathbf{13{:}02}, \mathbf{13{:}50}))]),\\ (k_2, [(v_2, [13{:}14, 13{:}44))])\end{array}$$

$$\downarrow \; GroupAlsoByWindow$$

$$\begin{array}{c}(k_1, [([\mathbf{v_1}, \mathbf{v_4}], [13{:}02, 13{:}50)),\\ ([\mathbf{v_3}], [13{:}57, 14{:}27))]),\\ (k_2, [([\mathbf{v_2}], [13{:}14, 13{:}44))])\end{array}$$

$$\downarrow \; ExpandToElements$$

$$\begin{array}{c}(k_1, [v_1, v_4], \mathbf{13{:}50}, [13{:}02, 13{:}50)),\\ (k_1, [v_3], \mathbf{14{:}27}, [13{:}57, 14{:}27)),\\ (k_2, [v_2], \mathbf{13{:}44}, [13{:}14, 13{:}44))\end{array}$$

TRIGGERS AND INCREMENTAL PROCESSING

Windowing: where in event time data are grouped

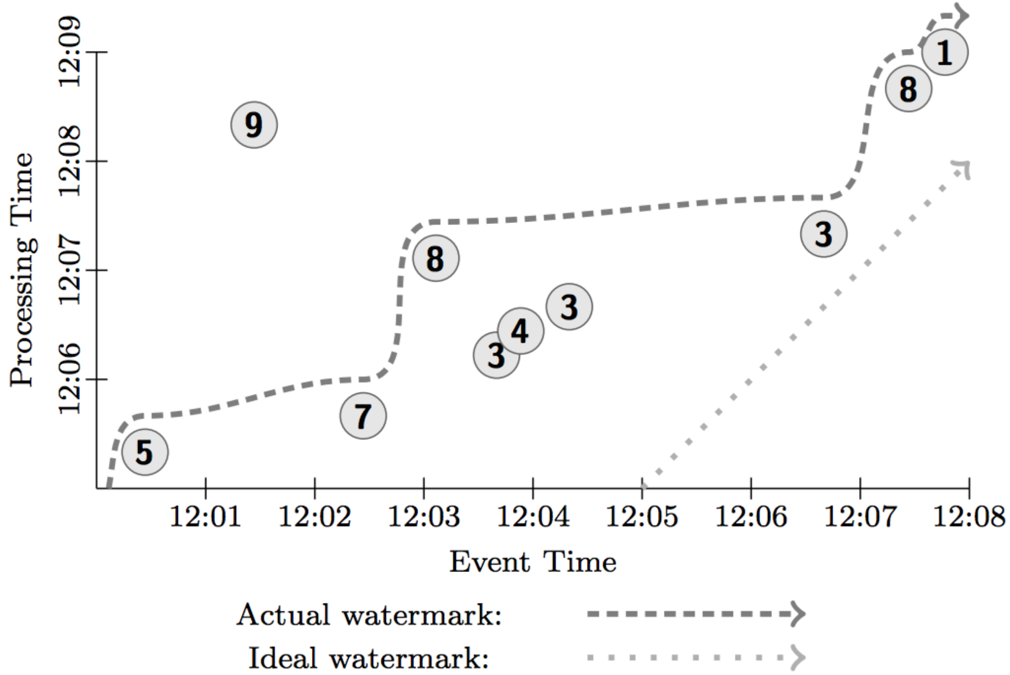
Triggering: when in processing time groups are emitted

Strategies

- Discarding
- Accumulating
- Accumulating & Retracting

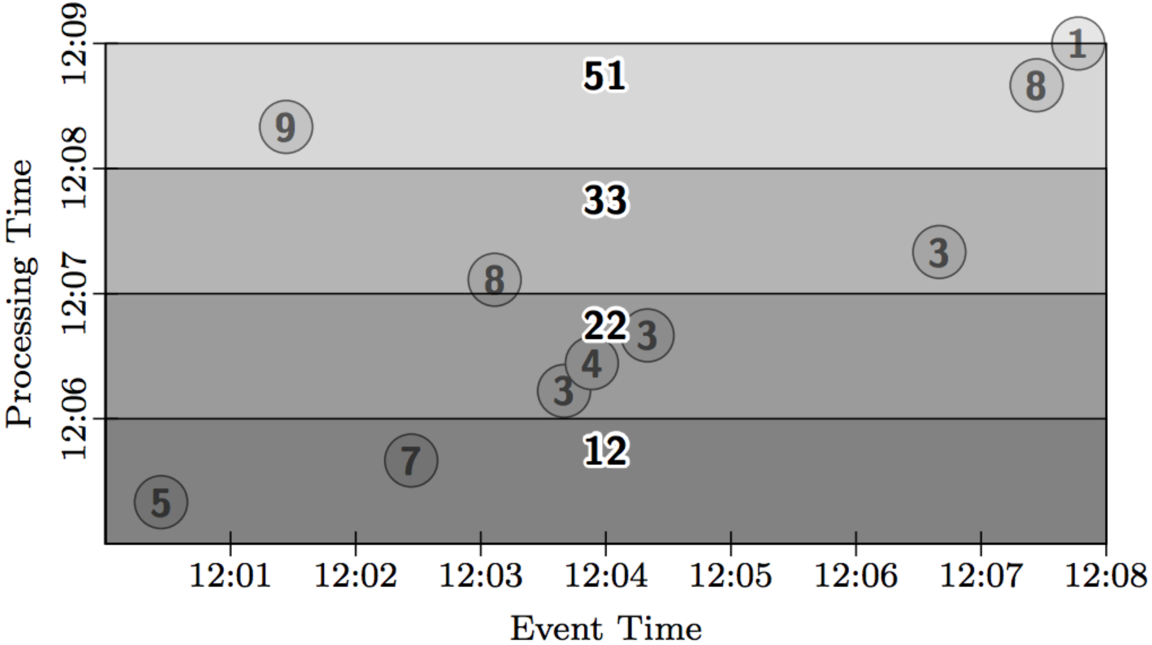
RUNNING EXAMPLE

```
PCollection<KV<String, Integer>> input = IO.read(...);
PCollection<KV<String, Integer>> output =
       input.apply(Sum.integersPerKey());
```



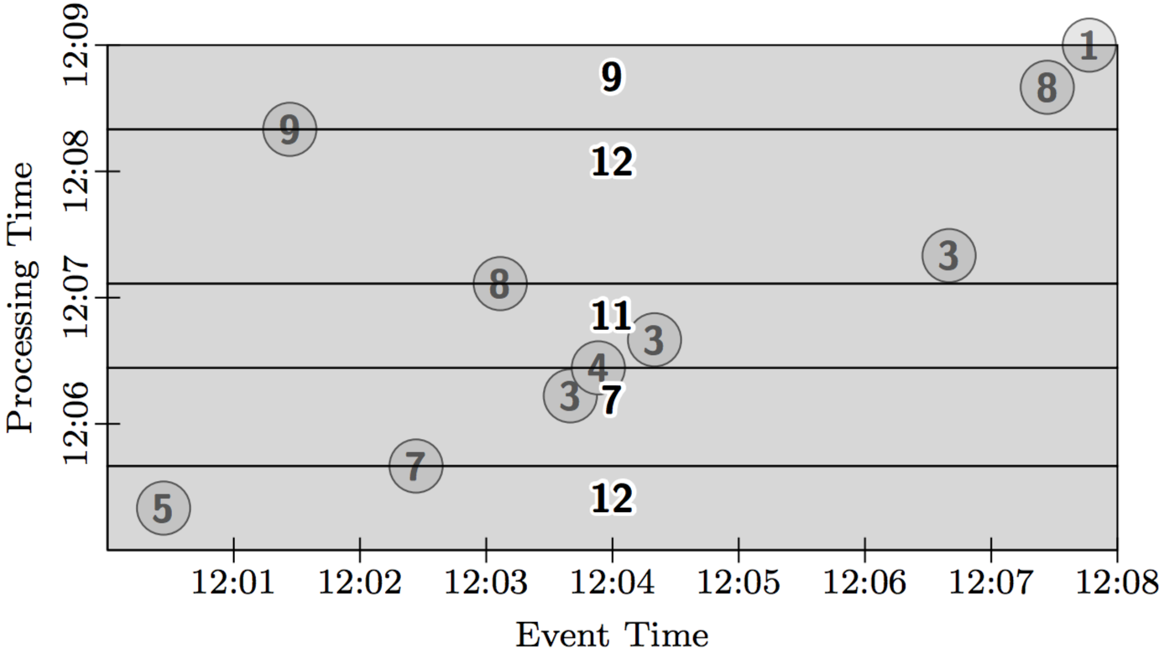
GLOBAL WINDOWS, ACCUMULATE

```
PCollection<KV<String, Integer>> output = input
    .apply(Window.trigger(Repeat(AtPeriod(1, MINUTE)))
                 .accumulating())
    .apply(Sum.integersPerKey());
```



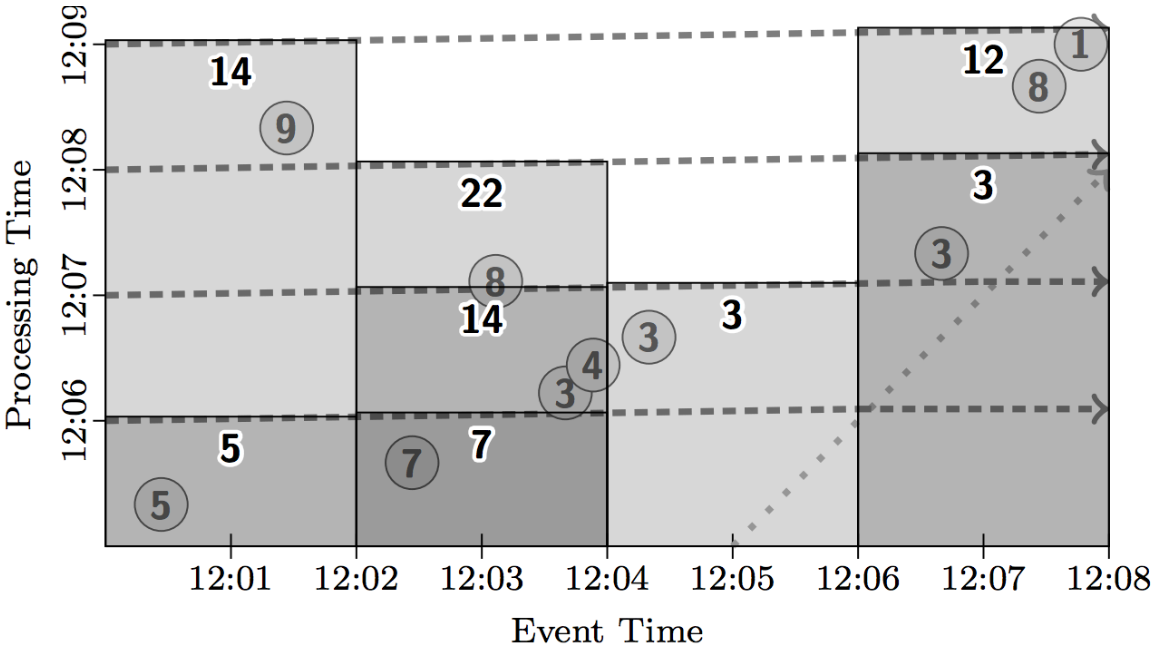
GLOBAL WINDOWS, COUNT, DISCARDING

```
PCollection<KV<String, Integer>> output = input
    .apply(Window.trigger(Repeat(AtCount(2)))
                 .discarding())
    .apply(Sum.integersPerKey());
```



FIXED WINDOWS, MICRO BATCH

```
PCollection<KV<String, Integer>> output = input
    .apply(Window.into(FixedWindows.of(2, MINUTES))
                 .trigger(Repeat(AtWatermark())))
                 .accumulating())
```



LESSONS / EXPERIENCES

Don’t rely on completeness

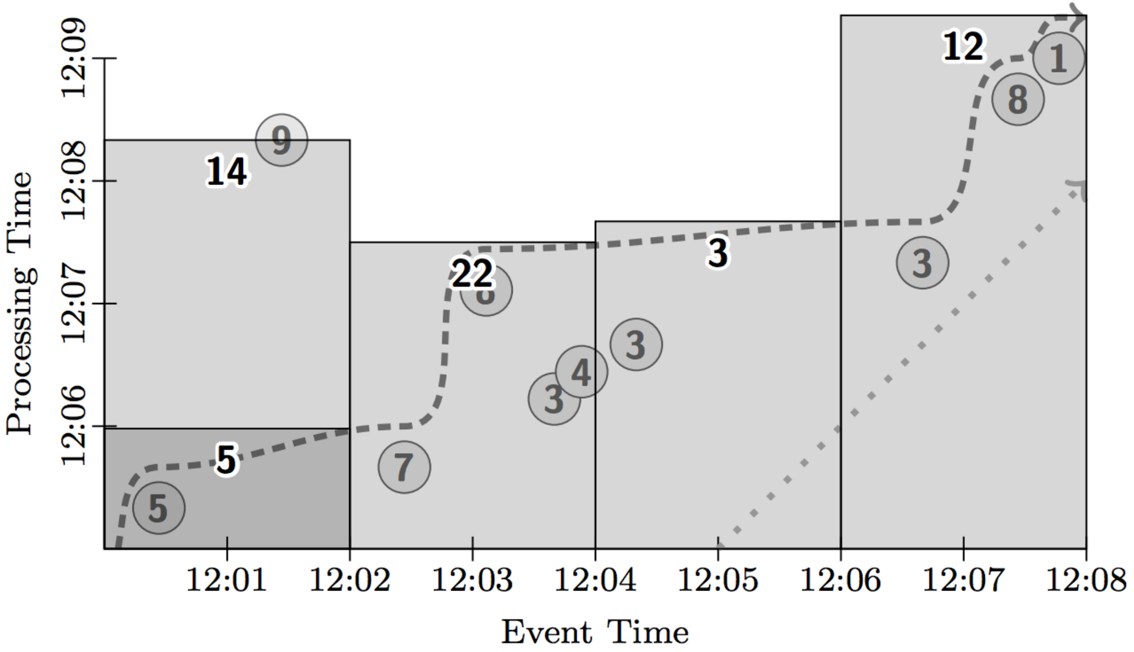
Be flexible, diverse use cases

- Billing
- Recommendation
- Anomaly detection

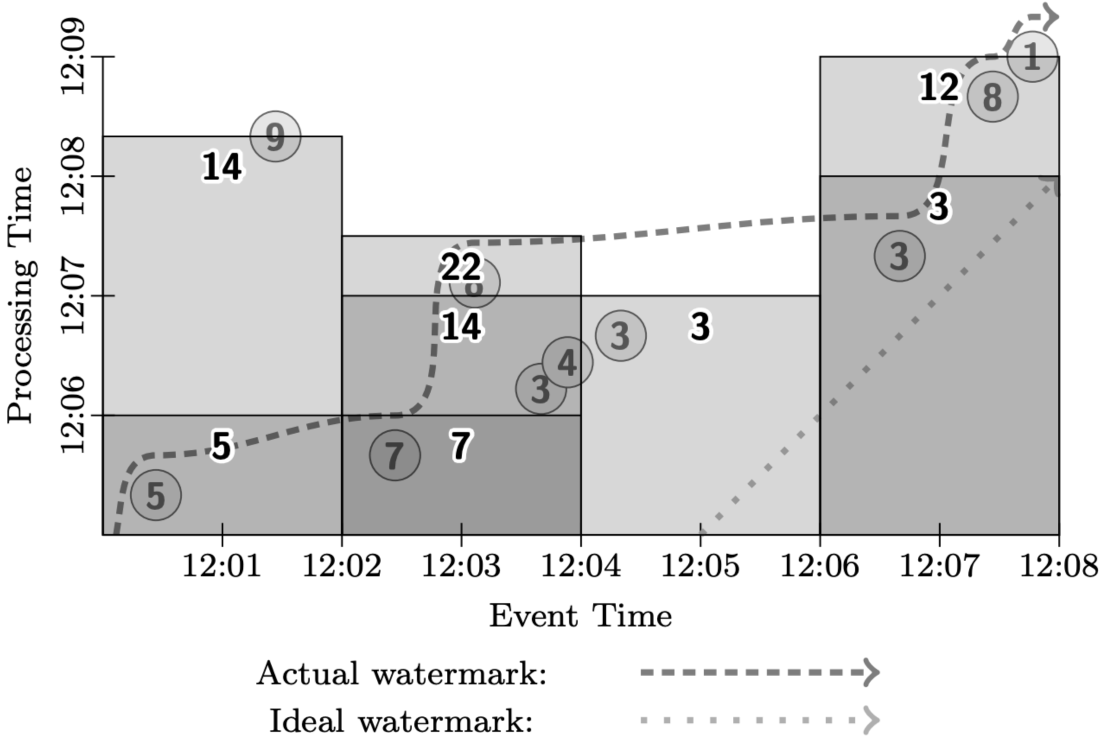
Support analysis in context of events

DISCUSSION

https://forms.gle/s7T2r67BDvkGQhmN9



Processing Time
12:06
12:07
12:08
12:09
Event Time
12:01
12:02
12:03
12:04
12:05
12:06
12:07
12:08
5
14
22
3
12
5
9
7
6
3
4
3
3
8
1



Processing Time
12:06
12:07
12:08
12:09
Event Time
12:01
12:02
12:03
12:04
12:05
12:06
12:07
12:08
14
9
5
5
22
8
14
7
7
3
4
3
3
3
3
12
8
1
Actual watermark:
Ideal watermark:

Consider you are implementing a micro-batch streaming API on top of Apache Spark. What are some of the bottlenecks/challenges you might have in building such a system?