Welcome!

CS 744: DATAFLOW

Shivaram Venkataraman

Fall 2020

ADMINISTRIVIA

- Assignment 2 grades are up! → Canvas
- Midterm grading in progress
- Course project proposal comments
  - ↳ Peer feedback   Thursday this week
  - ↳ Instructor feedback
- AEFIS feedback (next slide)

AEFIS FEEDBACK

Better organization

Improve writing on the slides, speak slower

Get a better internet connection? Better microphone?

Let me know how this sounds?

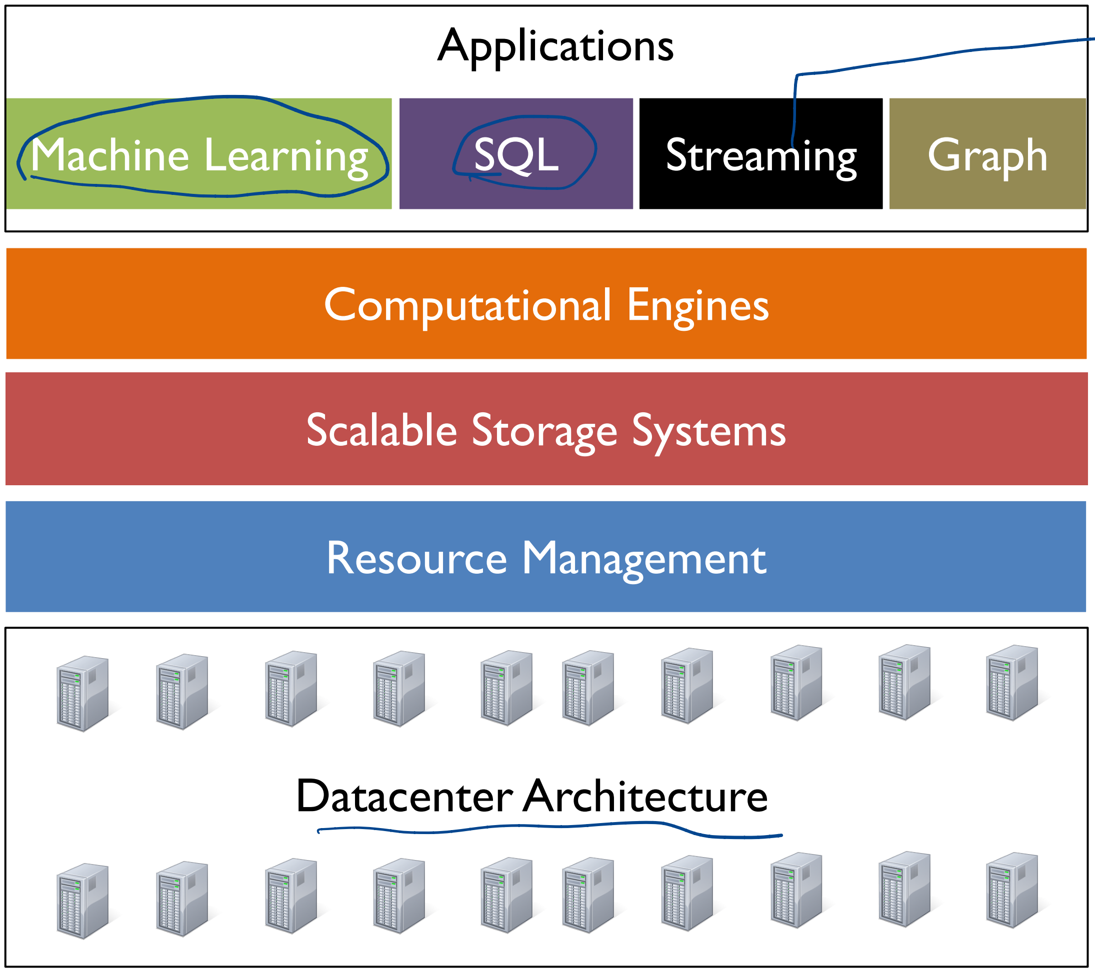
Piazza

More office hour slots

Discussion groups: same group each time? Also add prof. input

More time for Midterm exam, more guidance on deliverables

More homework/hands-on experience vs. too many evaluation components?



Stream Processing

MapReduce, Spark

GFS

Mesos
DRF

operators or DAG of operators

Spark SCOPE PyTorch

DATAFLOW MODEL (?)

MOTIVATION

ESPN. com

- Videos, each video has some ads

Streaming Video Provider

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

Dashboard
→ length of each session
→ which city etc.

→ Mobile phone Offline

Goals

- Easy to program
- Balance correctness, latency and cost

Unbounded data, out of order

how much delay till results are available

→ how accurate are your results

APPROACH

API Design → Dataflow Model

Separate user-facing model from execution

Decompose queries into

- What is being computed
- Where in time is it computed
- When is it materialized → Output
- How does it relate to earlier results

Developers writing applications

API

Batch

Streaming

(1) Framework processes bounded data

(2) MapReduce

(1) Framework can process data as it arrives

(2) similar to very small batch

Streaming

video viewing events

data

System

long running

output

process events as and when they arrive

Batch

events

Sys

Sys

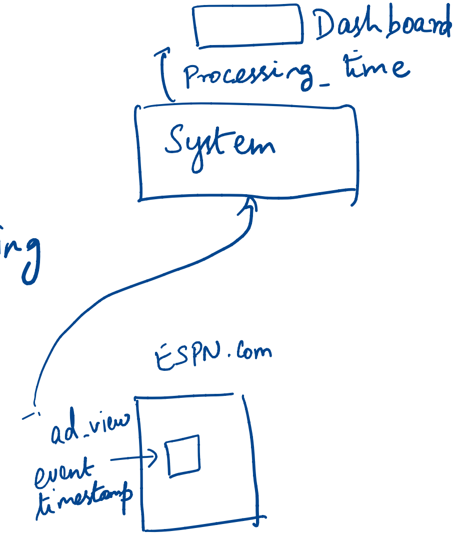
1 day

1 day

MR

MR

TERMINOLOGY



Unbounded/bounded data → Data is constantly arriving

Streaming/Batch execution

↳ see previous slide

Timestamps

Event time: Time when event occurs wrt user / input
e.g., time at ad was viewed in video

Processing time: Time at which an event is processed
e.g., time at which ad_view event is processed to update the dashboard.

WINDOWING

logical constructs

windows are aligned across keys

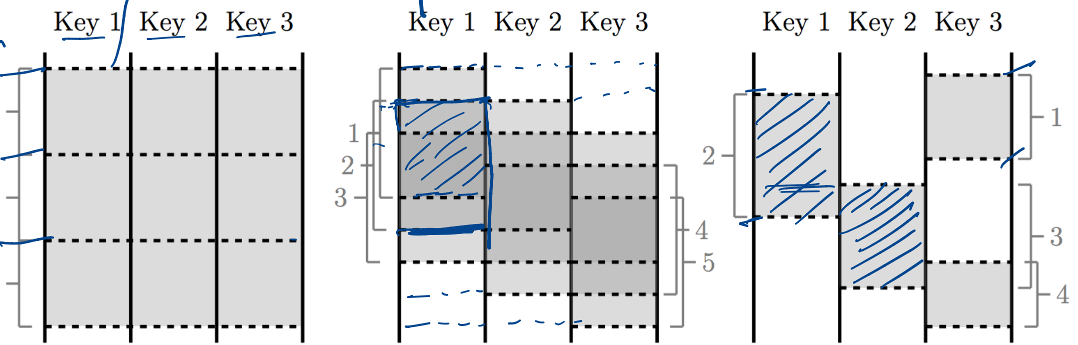
10 am to 11 am
10:30 am to 11:30 am
⋮

window_Id

Range of event_timestamp

10 am
11 am
12 pm

session length



Fixed or Tumbling windows

Do not overlap with each other

Sliding

overlap between consecutive windows

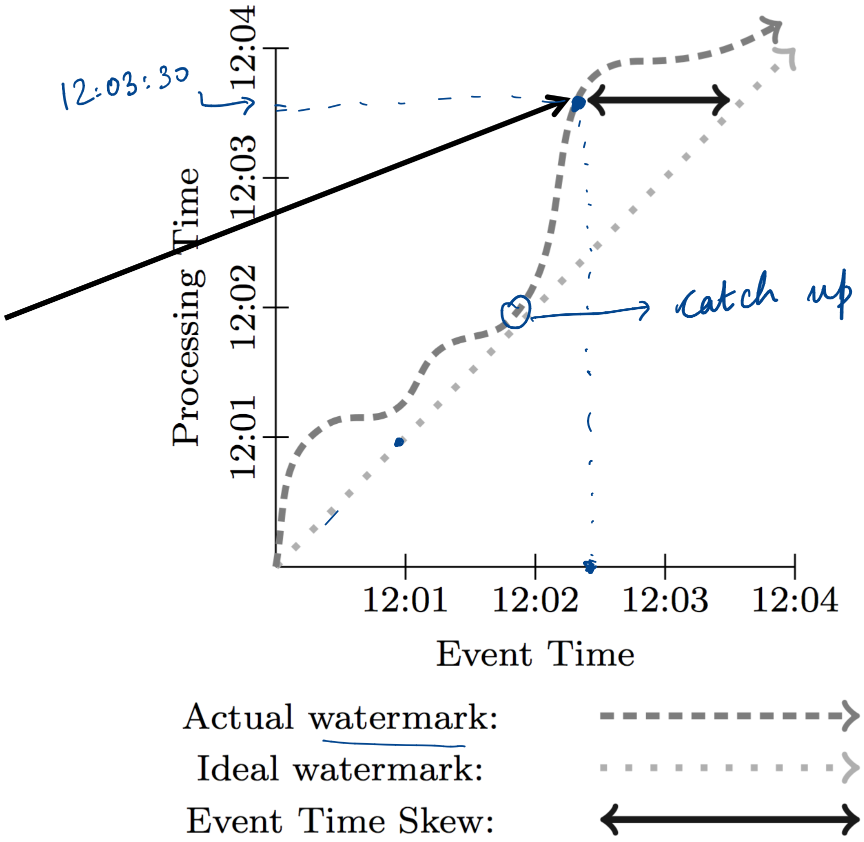
Sessions

Not aligned across all the keys

WATERMARK OR SKEW

- Watermark is not easy to know
- Heuristics
  - After 10 mins most devices send events

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



- Processing time lags event time
- Event time skew

No gap between event t & processing time

API

ParDo: Map in MapReduce or flatMap in Spark

GroupByKey: Reduce in MapReduce

Windowing

AssignWindow → Buckets tuple into a window

MergeWindow → Merge buckets based on strategy (sessions)

EXAMPLE

Assign tuples to Sessions

event_timestamp

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

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

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

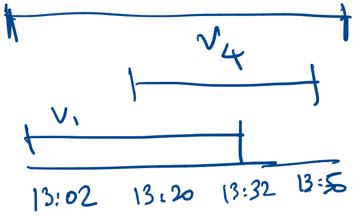
add 30m to event timestamp

$$\downarrow \; DropTimestamps$$

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

GroupByKey

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



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

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

overlap and merges them

$$\downarrow \; GroupAlsoByWindow$$

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

$$\downarrow \; ExpandToElements$$

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

TRIGGERS AND INCREMENTAL PROCESSING

Windowing: where in event time data are grouped

Triggering: when in processing time groups are emitted

Counter, sum of all views for a video

v1, 2
v1, 3
:
:
v1 6

$\boxed{2+3}$ → Output = 5

Output =

Strategies

- Discarding = 6
- Accumulating = 11
- Accumulating & Retracting

= -5, 11

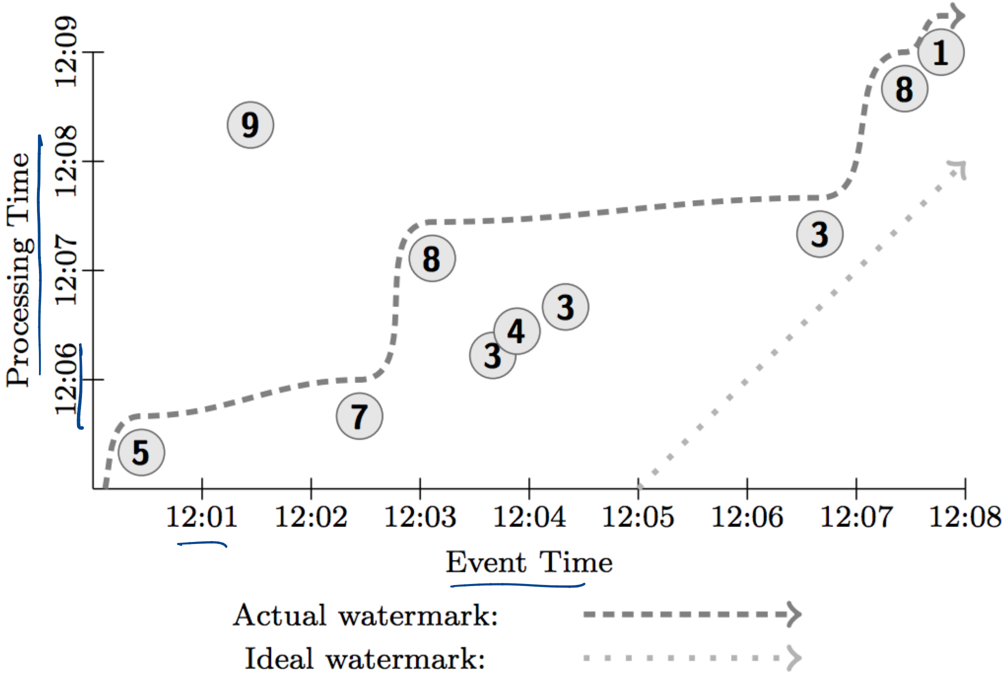
-5 ↓ retracting

11 ↓ Accumulating

RUNNING EXAMPLE

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

Sum of values for each key

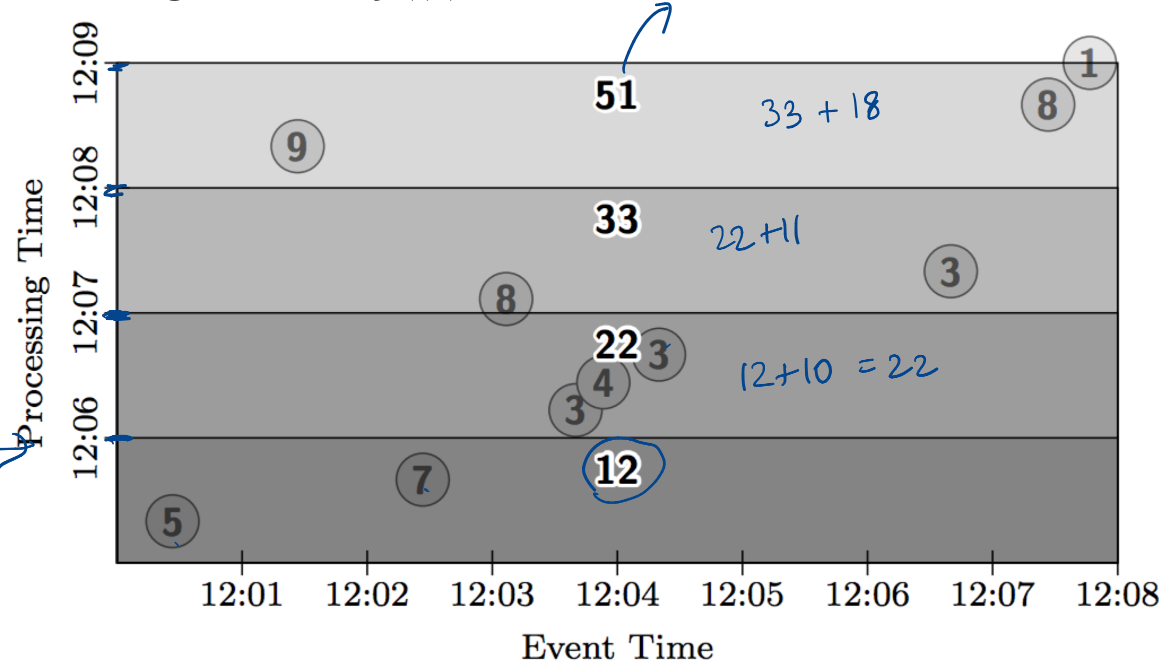


← Single key

GLOBAL WINDOWS, ACCUMULATE

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

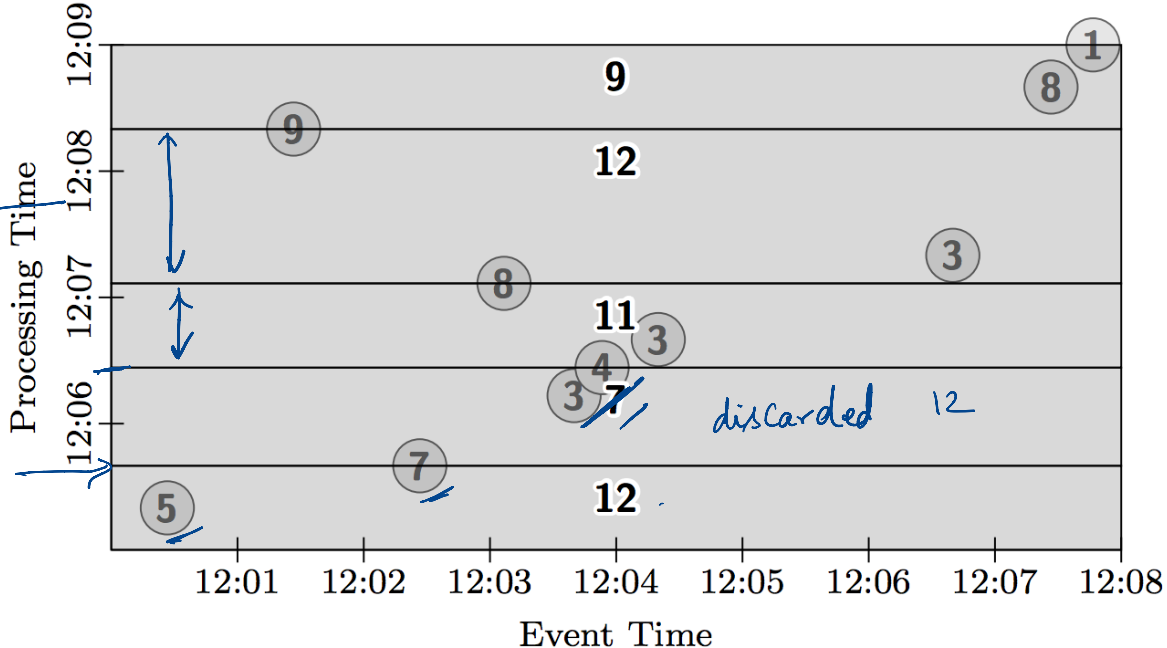
Triggered every 1 min in Processing time



GLOBAL WINDOWS, COUNT, DISCARDING

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

gap between outputs not same



FIXED WINDOWS, MICRO BATCH

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

12:00 - 12:02 5

12:02 - 12:04 ~~7~~ 14

12:04 - 12:06 3

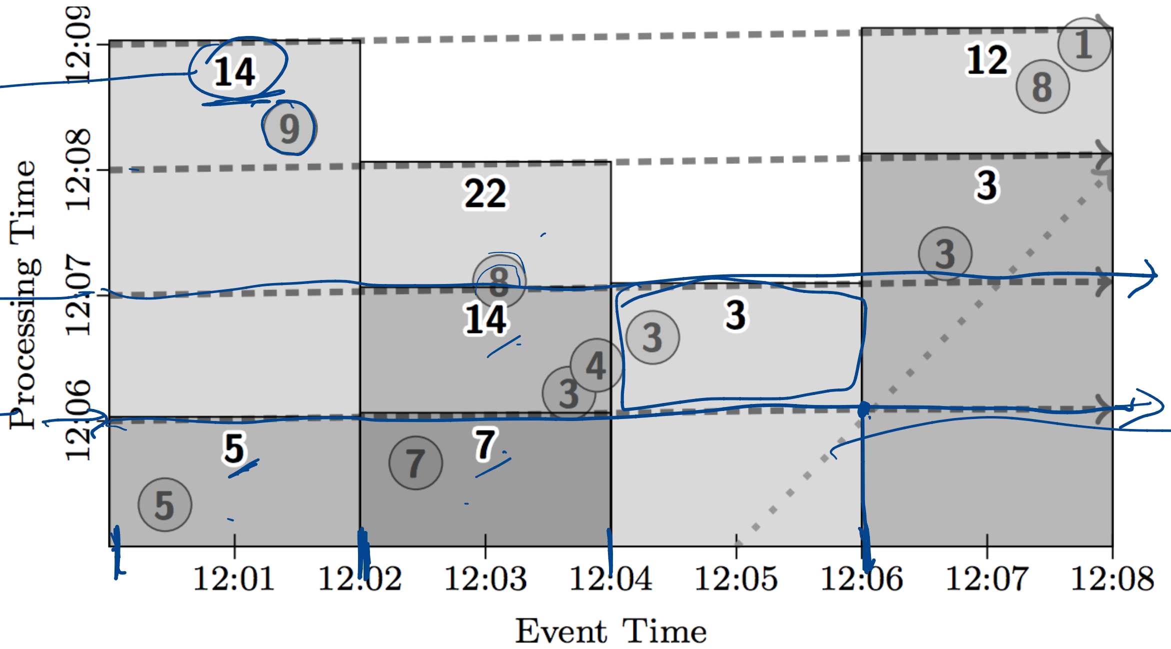
⋮

12:00 - 12:02 14

out of order

trigger every 1min

batch system runs



diagonal

x = y

SUMMARY/LESSONS

Design for unbounded data: Don't rely on completeness

Be flexible, diverse use cases

- Billing
- Recommendation
- Anomaly detection

Windowing, Trigger API to simplify programming on unbounded data

DISCUSSION

https://forms.gle/jwHjTBbR49vyQASq6

Fixed windows streaming

Assume watermark is given

(1) Window fires every time watermark pass

⇒ worse latency

⇒ fewer outputs

Stream

Batch

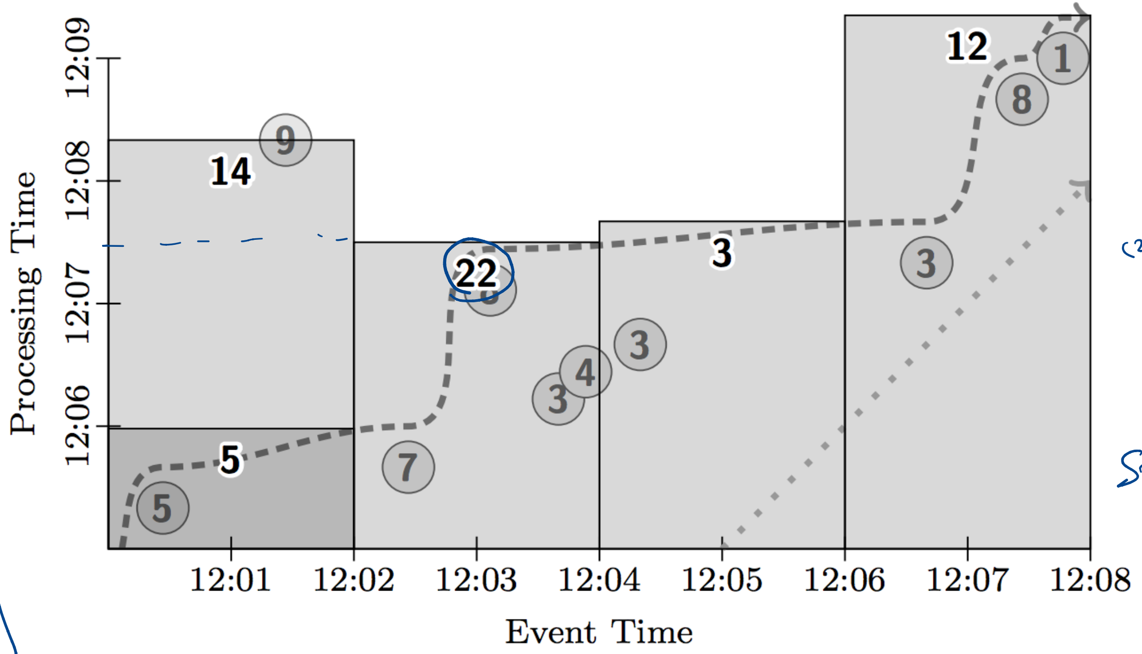
assign proc. time

Ingest

entry

Client

event_ts



(2) Micro batch partial results

Streaming ⇒ buffer events until watermark

→ Written to System

Apache Kafka

persist disk

Pub-Sub

Ingest time

proc_t

↳ Update query

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?

NEXT STEPS

Next class: Naiad

Course project proposal peer feedback