CS 744: BIG DATA SYSTEMS

Shivaram Venkataraman

Fall 2018

ADMINISTRIVIA

- Assignment 2 grades up, Midterm grades this week
- Course Projects: round 2 meetings (Sign up!)
- Next Tuesday: Guest speaker for first part

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

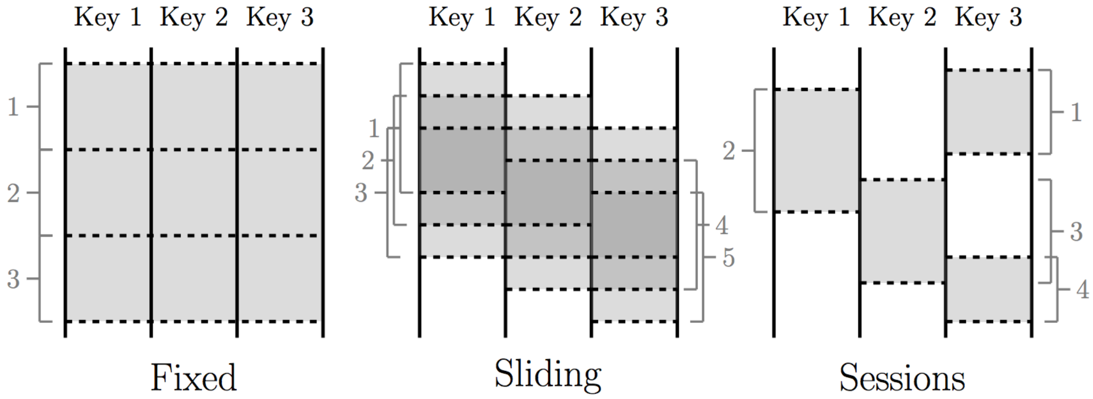
Unbounded/bounded data

- Streaming/Batch execution
- e.g., Flink vs. Spark Streaming

Timestamps

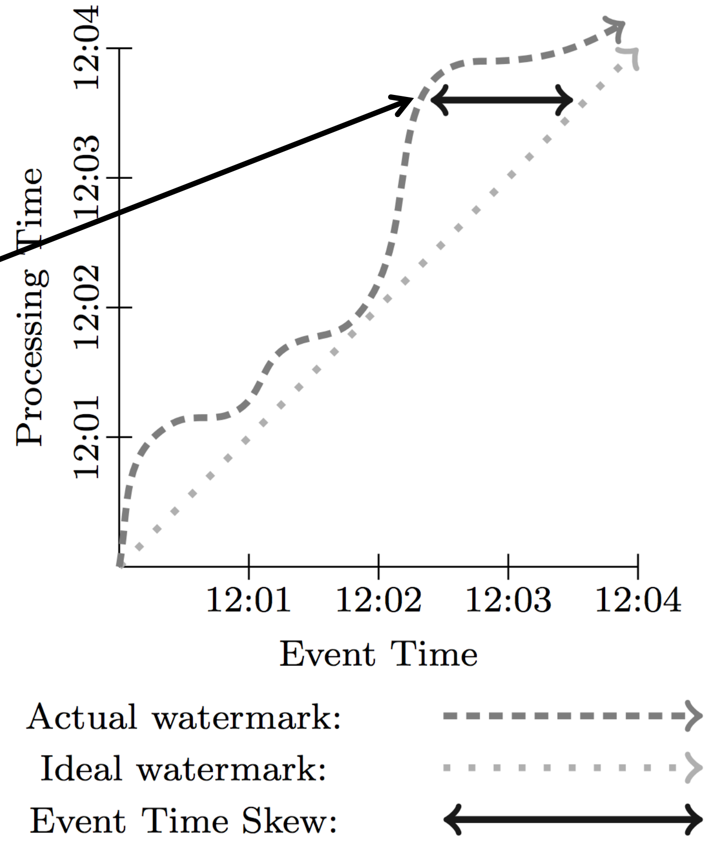
- Event time:
- Processing time:

Window types



WATERMARK OR SKEW

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



API

ParDo: Parallel Do (very similar to map or flatMap)

GroupByKey: Group values for a key

Windowing

AssignWindow – Bucket tuple as it arrives

MergeWindow – Merge buckets based on grouping strategy

EXAMPLE

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

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

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

$$\Big\downarrow \; DropTimestamps$$

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

GroupByKey

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

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

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

$$\Big\downarrow \; GroupAlsoByWindow$$

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

$$\Big\downarrow \; ExpandToElements$$

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

TRIGGERS AND INCREMENTAL PROCESSING

Windowing: where in event time data are grouped

Triggering: when in processing time groups are emitted

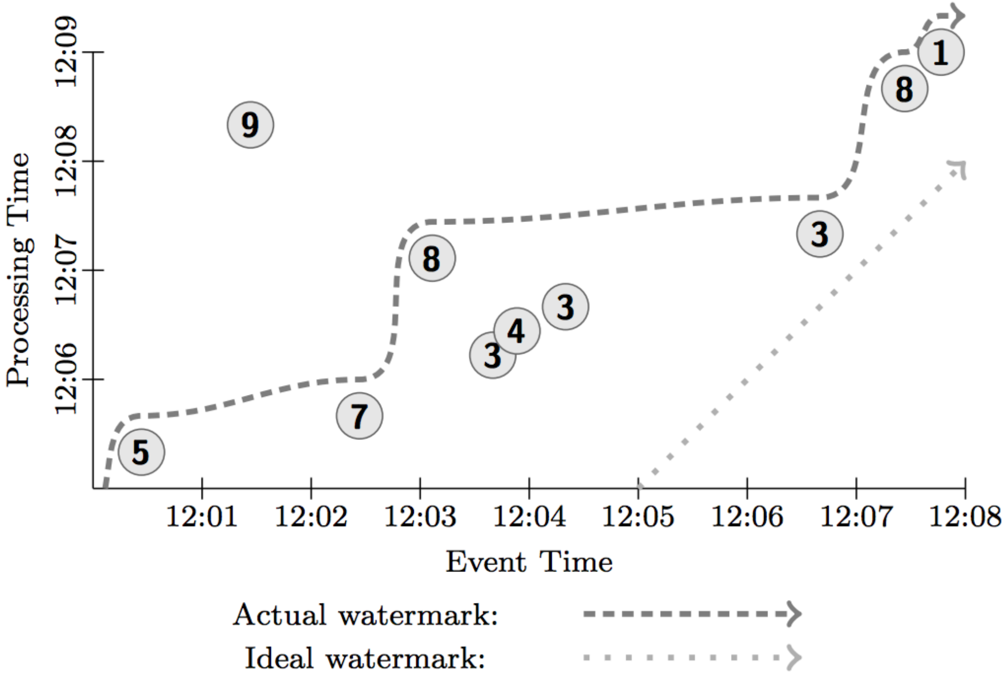
Strategies

- Discarding
- Accumulating
- Accumulating & Retracting

Details with running example

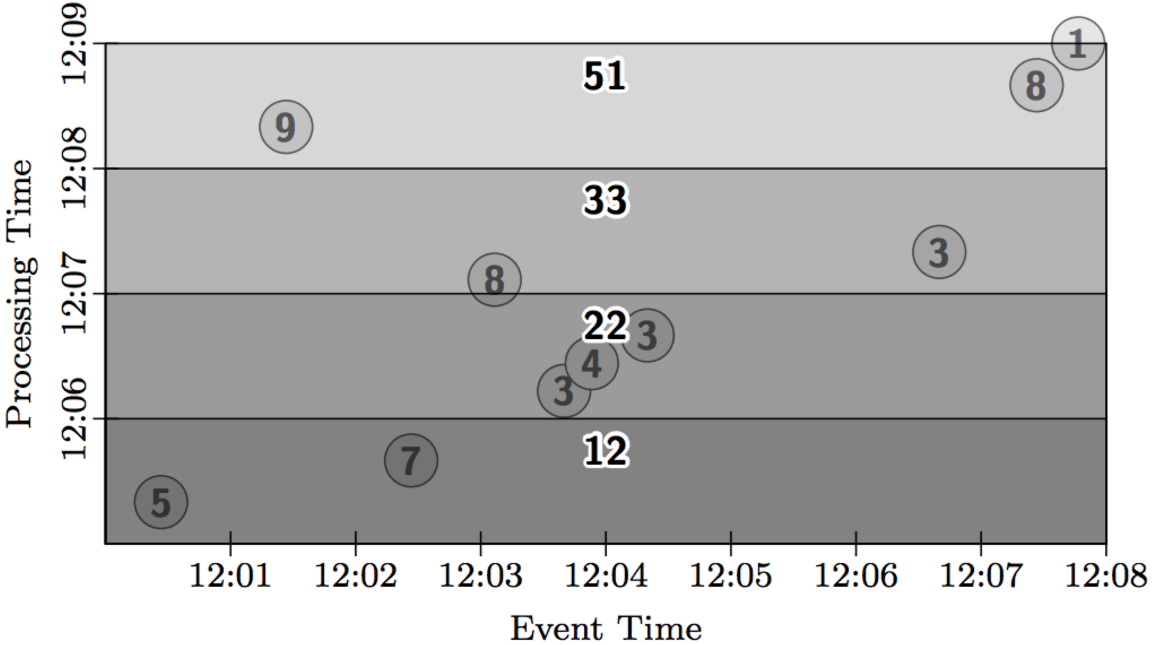
RUNNING EXAMPLE

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



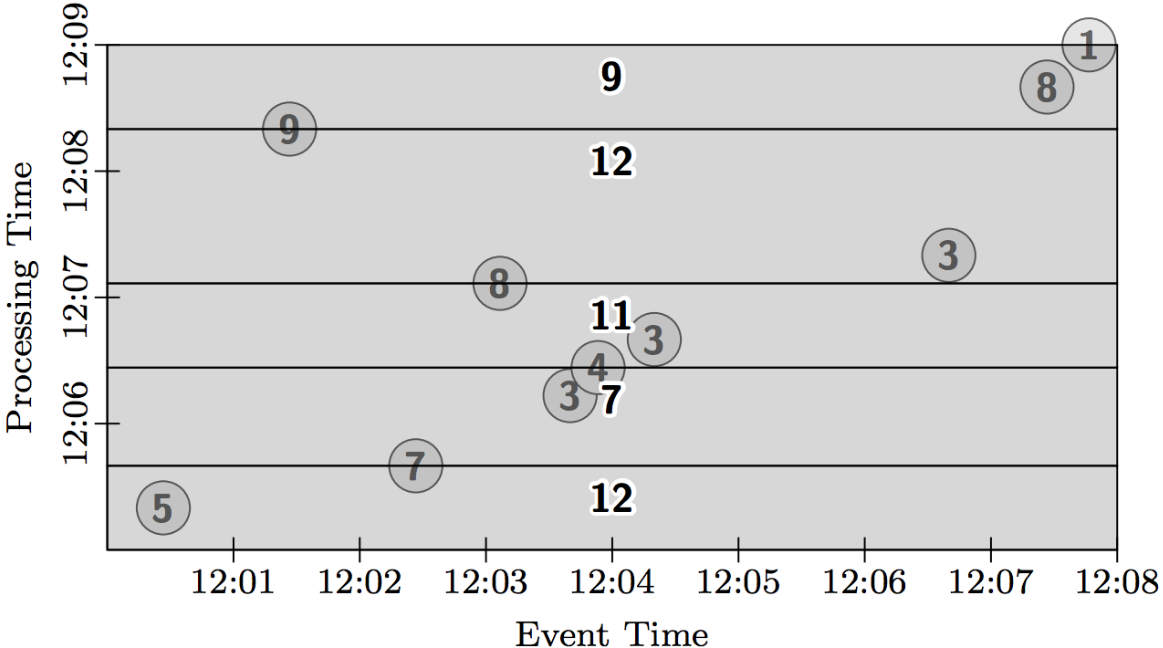
GLOBAL WINDOWS, ACCUMULATE

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



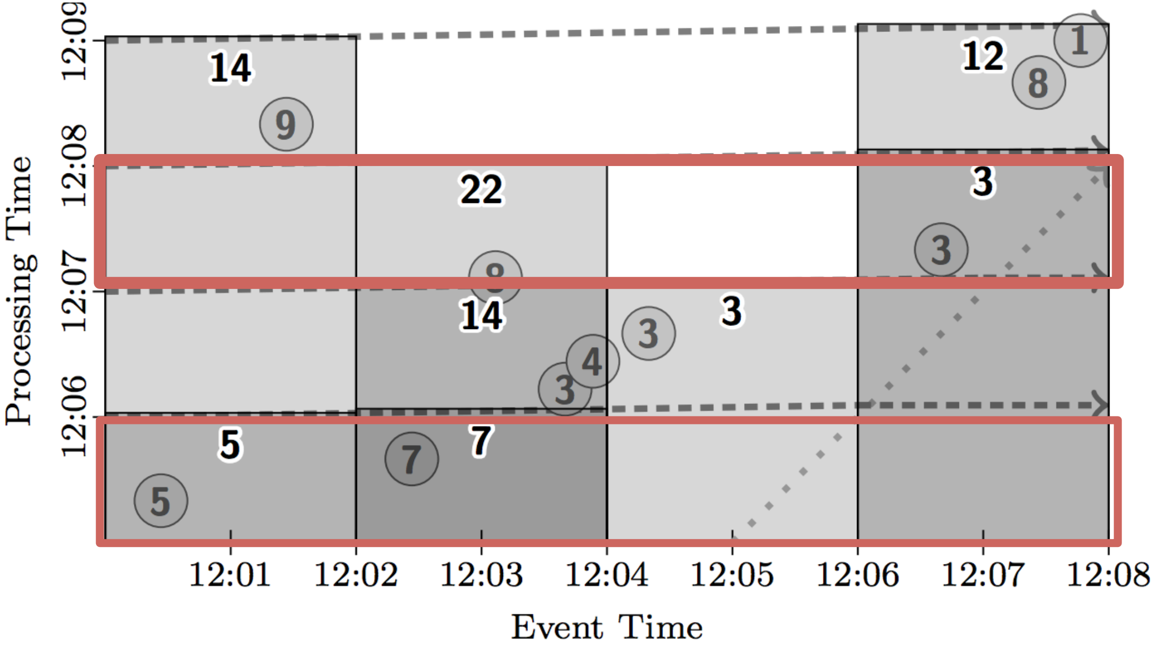
GLOBAL WINDOWS, COUNT, DISCARDING

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



FIXED WINDOWS, MICRO BATCH

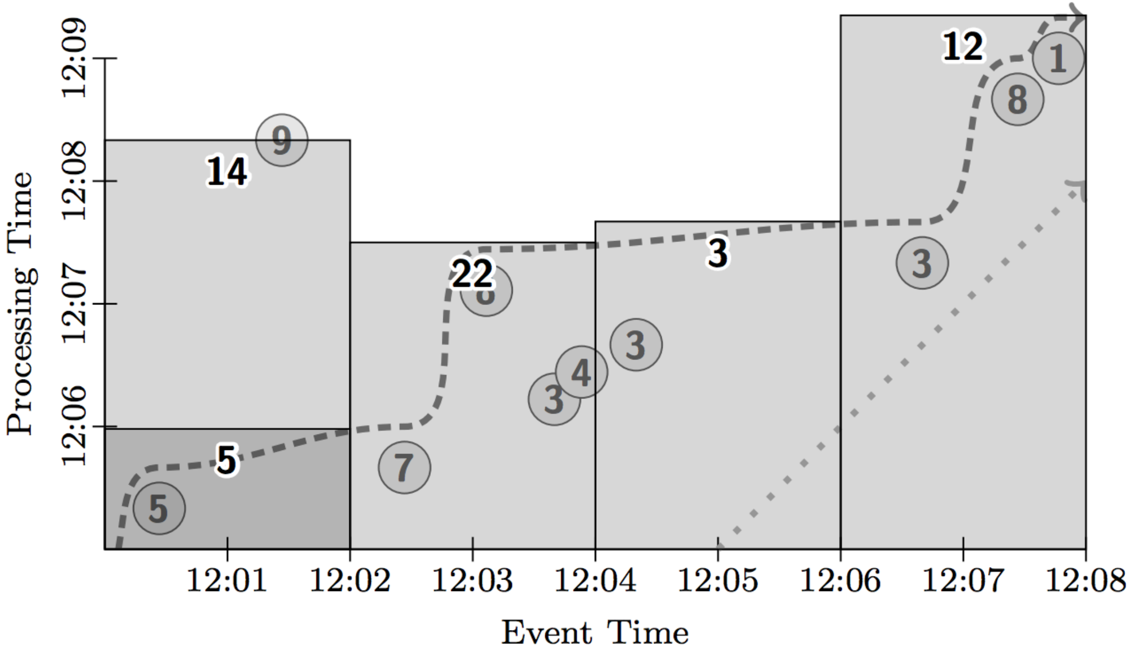
```
PCollection<KV<String, Integer>> output = input
    .apply(Window.into(FixedWindows.of(2, MINUTES))
                 .trigger(Repeat(AtWatermark())))
                 .accumulating())
```



FIXED WINDOWS, STREAMING

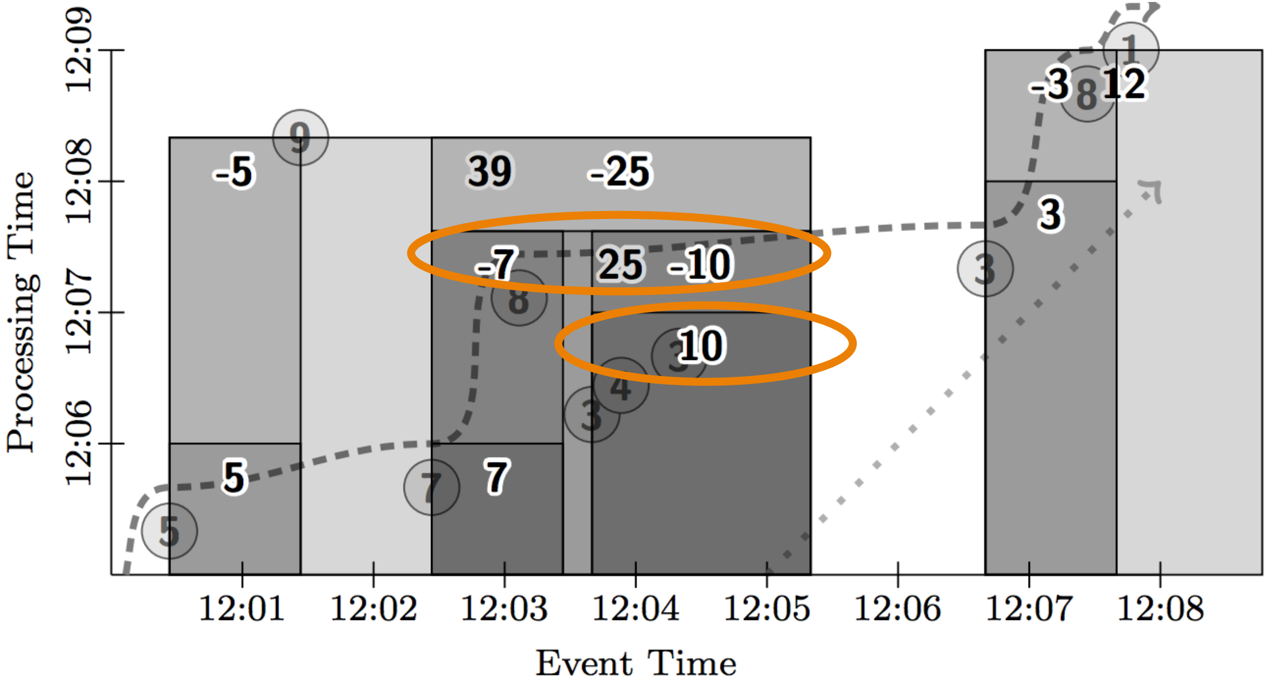
```
PCollection<KV<String, Integer>> output = input
    .apply(Window.into(FixedWindows.of(2, MINUTES))
                 .trigger(Repeat(AtWatermark())))
                 .accumulating())
```

Option to repeat at processing time intervals



SESSIONS, RETRACTING

```
output = input.apply(
Window.into(Sessions.gap(1,Min))
  .trigger(SequenceOf(
      RepeatUntil(
        AtPeriod(1, MINUTE),
        AtWatermark()),
      Repeat(AtWatermark())))
  .accumulatingAndRetracting(
  .apply(Sum.integersPerKey()
```



LESSONS / EXPERIENCES

- Don’t rely on completeness

- Be flexible, diverse use cases
  - Billing
  - Recommendation
  - Anomaly detection

- Support analysis in context of events

SUMMARY

- Model of streaming window triggers, processing
- Separate user-level view from implementation
- Motivated by real-world use cases, Cloud Dataflow SDK