- Assignment 2 grades up
- Midterm grading
- Course project proposal comments
- AEFIS feedback
- No Class next Tuesday?
DATAFLOW MODEL (?)

File

Data

Run

Map Reduce

Sparks

Output

Static dataset

Requests

Services

Logaties

Stream dataset

user1: <videos user has watched>

Correctness vs. latency
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: Time when event occurred w/ user input

Processing time: Time at which event is processed

All of these have proc. time = 199
WINDOWING

- **Fixed**
  - No overlap
  - Duration

- **Sliding**
  - An overlap
  - Slide duration

- **Sessions**
  - Window size
  - Specific for key
  - Unaligned

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

![Diagram showing actual and ideal watermarks with event time skew]
API

ParDo:  
`map` | `flatMap` | `Map`

GroupByKey:  
`reduce`

Windowing

AssignWindow  →  Bucket tuples into window

MergeWindow  →  Merge buckets based on strategy
EXAMPLE

AssignWindows(Sessions(30m))

(k1, v1, 13:02, [13:02, 13:32]),
(k2, v2, 13:14, [13:14, 13:44]),
(k1, v3, 13:57, [13:57, 14:27]),
(k1, v4, 13:20, [13:20, 13:50])

DropTimestamps

(k1, v1, [13:02, 13:32]),
(k2, v2, [13:14, 13:44]),
(k1, v3, [13:57, 14:27]),
(k1, v4, [13:20, 13:50])

GroupByKey

MergeWindows(Sessions(30m))

(k1, [(v1, 13:02, 13:32)],
(v3, [13:57, 14:27]),
(v4, [13:20, 13:50]),
(k2, [(v2, [13:14, 13:44])])

GroupAlsoByWindow

(k1, [(v1, v2, [13:02, 13:50]),
(v3, [13:57, 14:27])],
(k2, [v2, [13:14, 13:44]])

ExpandToElements

(k1, [v1, v4, 13:50, [13:02, 13:50]),
(k1, v3, 14:27, [13:57, 14:27]),
(k2, [v2, 13:44, [13:14, 13:44]])
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());
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
Both worlds
more computation than just watermark

Trigger is composite

Time + watermark

Latency?

Partial results

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?