ADMINISTRIVIA

- Assignment 2 grades are up!
- Midterm grading in progress
- Course project proposal comments

- AEFIS feedback (next slide)
Improve writing on the slides, speak slower
Get a better internet connection? Better microphone?

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?
DATAFLOW MODEL (?)
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
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

Fixed

Sliding

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

ParDo:

GroupByKey:

Windowing
  AssignWindow

MergeWindow
\[(k_1, v_1, 13:02, [0, \infty)),
(k_2, v_2, 13:14, [0, \infty)),
(k_1, v_3, 13:57, [0, \infty)),
(k_1, v_4, 13:20, [0, \infty))\]

\[\text{AssignWindows}(\text{Sessions}(30m))\]

\[(k_1, v_1, 13:02, [13:02, 13:32)),
(k_2, v_2, 13:14, [13:14, 13:44)),
(k_1, v_3, 13:57, [13:57, 14:27)),
(k_1, v_4, 13:20, [13:20, 13:50))\]

\[\text{DropTimestamps}\]

\[(k_1, v_1, [13:02, 13:32)),
(k_2, v_2, [13:14, 13:44)),
(k_1, v_3, [13:57, 14:27)),
(k_1, v_4, [13:20, 13:50))\]

\[\text{GroupByKey}\]

\[(k_1, [(v_1, 13:02, 13:32)),
(v_3, [13:57, 14:27)),
(v_4, [13:20, 13:50))]),
(k_2, [(v_2, [13:14, 13:44)])]\]

\[\text{MergeWindows(Sessions}(30m))\]

\[(k_1, [(v_1, 13:02, 13:50)),
(v_3, [13:57, 14:27)),
(v_4, [13:02, 13:50))]),
(k_2, [(v_2, [13:14, 13:44)])]\]

\[\text{GroupAlsoByWindow}\]

\[(k_1, [[(v_1, v_4), 13:02, 13:50)),
((v_3), [13:57, 14:27))]),
(k_2, [[(v_2), [13:14, 13:44)])]\]

\[\text{ExpandToElements}\]

\[(k_1, [v_1, v_4, 13:50, [13:02, 13:50)),
(k_1, [v_3, 14:27, [13:57, 14:27)),
(k_2, [v_2, 13:44, [13:14, 13:44))]\]
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());
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());
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
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