CS 744: DATAFLOW

Shivaram Venkataraman
Fall 2020
- 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?

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?

Let me know how this sounds?
DATAFLOW MODEL (?)

operators or DAG of operators

Spank  Scope  PyTorch
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 → 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
- How does it relate to earlier results
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 which 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

Do not overlap with each other

Fixed or Sliding Sessions

overlap between consecutive windows

not aligned across all the keys

session length
WATERMARK or SKEW

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

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

Processing Time

12:01 12:02 12:03 12:04

Event Time

12:01 12:02 12:03 12:04

Actual watermark:
Ideal watermark:
Event Time Skew:

- Processing time lags event time
- Event time skew

No gap between event time & processing time
API

ParDo: \quad \text{Map in MapReduce or FlatMap in Spark}

GroupByKey: \quad \text{Reduce in MapReduce}

Windowing

AssignWindow \quad \Rightarrow \quad \text{Buckets tuple into a window}

MergeWindow \quad \Rightarrow \quad \text{Merge buckets based on strategy (sessions)}
Assign tuples to sessions:

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

\[\downarrow\]

**AssignWindows**

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

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

\[\downarrow\]

**DropTimestamps**

\[
\text{GroupByKey}
\]

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

\[\downarrow\]

**MergeWindows**

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

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

\[\downarrow\]

**GroupAlsoByWindow**

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

\[\downarrow\]

**ExpandToElements**

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

\[ \text{Output} = -5, 11 \]

\[ \text{Output} = \]

Counter, sum of all views for a video

\[ \text{Output} = 5 \]
RUNNING EXAMPLE

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

- Single key
- Sum of values for each key
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\langle KV< String, Integer >> output = input
    .apply(Window.into(FixedWindows.of(2, MINUTES)))
    .trigger(Repeat(AtWatermark()))
    .accumulating()
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

(2) Micro batch partial results
   Streaming buffer events until watermark

stream ⏳ batch ⏳ assign proc. time

Ingest entry

Client entry-pts

Written to System
Apache Kafka persist disk

Pub-Sub Ingest proc-\_t 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