Streaming Systems

Chapter 2
The What, Where, When, and How of Data Processing

2.1

- Trigger:
  - When the output for a window should be materialized/emitted. May have multiple triggers within one window
- Watermarks:
  - input completeness with respect to event times. A watermark with value of time X makes the statement: “all input data with event times less than X have been observed.”
- Accumulation:
  - specifies the relationship between multiple results that are observed for the same window.
  - 1. Discard. 2. Accumulate: simply overwrite
  - 3. a & retract: both new value and retracted value visible to downstream consumer
    - Helpful when: the new value can’t just overwrite the old value; you instead need the retraction to remove the old value . Or , the new value might be replacing more than one previous window, due to window merging

2.1

- What results are calculated?
    - transformations
- Where in event time are results calculated?
    - windowing
- **When in processing time are results materialized?**
    - Triggers (happen in processing time) plus watermarks (progressing in event time domain)
- How do refinements of results relate?
    - accumulation

2.2

- Trigger:
  - Pane: Each specific output for a window is referred to as a pane of the window.
  - Repeated update triggers (multiple times)
    - How to decide the intervals:
    - aligned delays (like microbatch):
      - Upside: regular updates across all modified windows; good predictability
      - Downside: all updates happen at once, bursty workloads
    - unaligned delays: more even load dist. over time
  - Completeness triggers (more closely align with classic batch processing. Less used)

2.2

- Watermarks:
  - it captures the progress of event-time completeness as processing time progresses. Conceptually, you can think of the watermark as a function, F(P) → E, which takes a point in processing time and returns a point in event time.
  - Reason about the completeness of our input, for uses cases that wants to reason lack/missing data (like outer join, anomaly detection)
  - Low watermarks here (spark streaming: high optimistic watermarks):
    - **Perfect watermarks**: all inputs with event times less than E have been observed.
    - **Heuristic watermarks**:
      - Watermark algorithm in use is independent from the pipeline itself
      - Remarkably accurate. May have late data. Need to deal with late data
  - Downside: too slow + too fast?

2.2

- Watermarks (cont.)
  - Too slow to reason about completeness due to unprocessed lagging data
    - Not good for latency. Maybe better to refine overtime later and eventually complete
  - Too fast maybe for a heuristic watermark: incorrectly advanced earlier than should be
    - Late data. If discard, will impact correctness
- Design
  - Trigger: provide low-latency updates but no way to reason about completeness
  - Watermark: provide a notion of completeness but variable and possible high latency
  - Why not combine both! =>
    - Early/on-time/late triggers
    - Allowed lateness for garbage collection of heuristic watermark (not for perfect w*)

2.2

- When: Early/On-Time/Late Triggers FTW!
  - Zero or more early panes: This compensates for watermarks sometimes being too slow
  - A single on-time pane
  - Zero or more late panes: This compensates for watermarks being too fast
- Design:
  - These new triggers is that they effectively normalize the output pattern between the perfect and heuristic watermark versions.
  - The biggest remaining difference between the perfect and heuristic early/ontime/late versions at this point is window lifetime bounds.
- Heuristic EOL triggers:
  - have some delay/lateness to account for late data.
  - Set a horizon in event-time domain. NOT processing time (avoid system bugs).
  - Perfect triggers: horizon sets to 0