Streaming Data Engineering and (some) ML

intro

Ankit Virmani

- ~ 10 years in Big Data Engineering, Data Governance, AI/ML
- Recently focused on responsible AI, real-time data and linear programming
- Seattle is home
- Simple passions in life: Dogs, Horror movies and Data

agenda

- Core Terms in Streaming that are different from batch processing
- Technical Challenges in streaming data and how to address
- Example use Case: Fraud Detection using batch and streaming- highlighting the differences
- How do we think about data governance in streaming data pipelines

some terms used in streaming data

event time : time at which an event actually occurred

processing time : time at which the event was/will be processed by the streaming engine

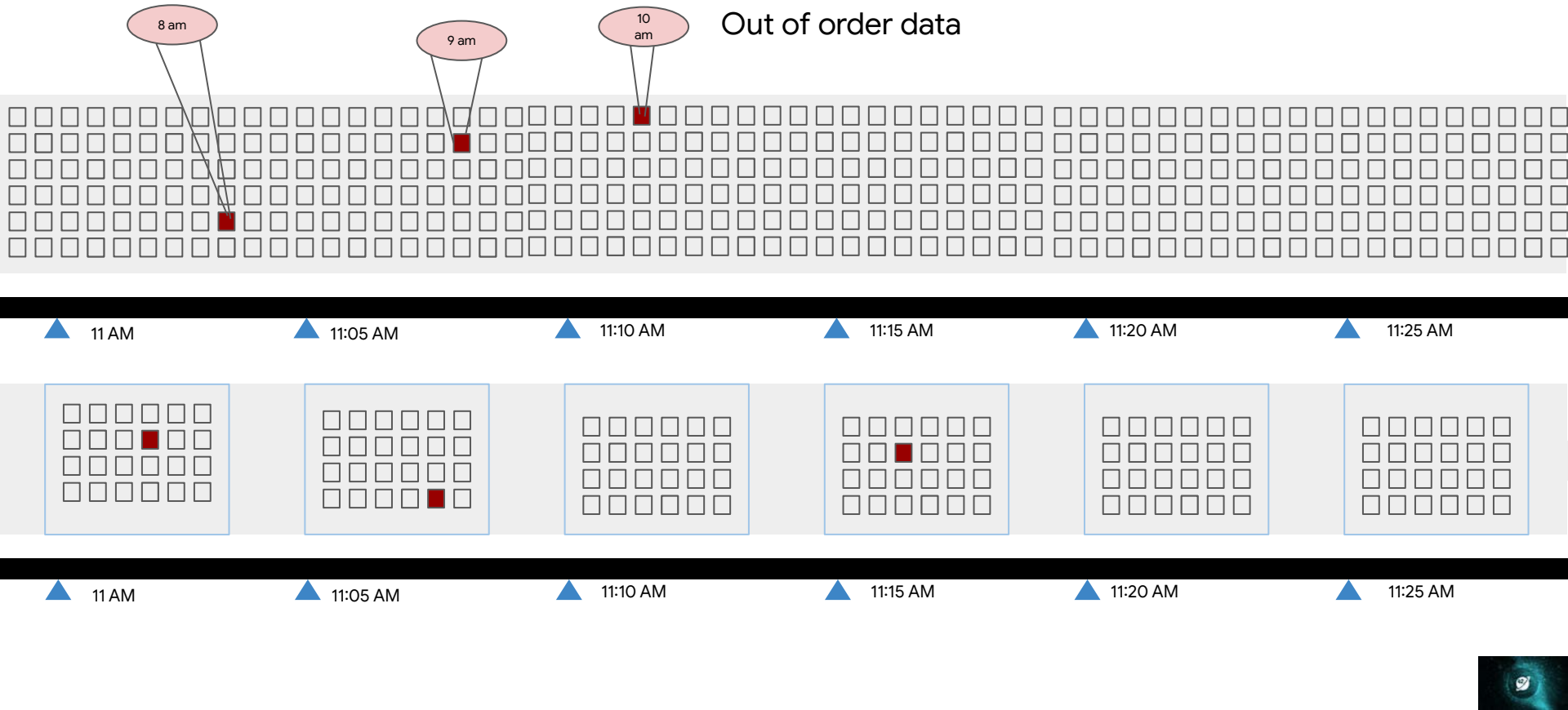
watermark : notion of completeness

window : chopping up the data into temporal boundaries for groupings and aggregations

trigger : when in the processing times are the results materialized

accumulation : relationship between the multiple results within the same window

some interesting challenges in streaming



dilemma between correctness and completion

correctness:
batch- I process the bounded data based on the business logic
streaming - I process the unbounded data based on the business logic + figure how to handle the out of order data

completion:
batch- I process all the records in a bounded dataset (file, table etc)
streaming - I try to process all data- on time and late arriving

Watermark	Trigger	Accumulation
Notion of completeness All input data with event time less than X has been processed Not perfect, need strategy to process late arrivals	When to emit the aggregated results for a window Event time, processing time, or data driven triggers	Accumulation mode determines whether the system accumulates the window panes or discards them.

bringing all the concepts together

What results are being calculated	When in the event time are they being calculated	When in the processing time are the results materialized
Transformations Filters Aggregations ML training SQL	Event-time windowing (Similar to sharding in batch) Fixed Sliding Session	Watermarks+Triggers How do the refinements of results relate: Accumulation Late Arrivals: Triggers

```
word_lengths = words | beam.FlatMap(lambda
word: [len(word)])
```

```
from apache_beam import window
fixed_windowed_items = (
   items | 'window' >>
beam.WindowInto(window.FixedWindows(60)))
```

```
pcollection | WindowInto(
   FixedWindows(1 * 60),
   trigger=AfterProcessingTime(1 * 60),
accumulation_mode=AccumulationMod
e.DISCARDING)
```

streaming at Scale: Infrastructure and code considerations

Autoscaling: Vertical and Horizontal	Horizontal and vertical autoscaling (in flight)
Dynamic Work Rebalancing	If you can't avoid stragglers, use the infrastructure that provides dynamic work rebalancing
Window processing decoupling	Decoupling window processing from other stream operations helps scale better
Right Fitting	Give memory and GPU related resource hints for a particular pipeline or specific steps in the pipeline to optimize performance
Shuffle reduction	Combineby: reduced shuffle, groupby: increased shuffle Use combineby wherever possible
Retrying forever- NO	Always have time period or retry count, which returns an error and ideally sends the element to a dead letter queue
FileIO	Provide appropriate # shards while writing to balance parallelism and output file sizes, anywhere from 100 MB to 1 GB per shard, depending on the total size of the output data.
Network	Keep all the sources and sinks in the same region (if possible) to reduce network latency.
Type hints	When you use type hints, Beam raises exceptions during pipeline construction time, rather than runtime.- helps catch errors much sooner than being deep in the pipeline run