DATA516/CSED516
Scalable Data Systems and Algorithms

Lecture 8
Stream Processing and Review

Administrivia

- **Deadlines**
  - HW3 Due: **Friday, December 8th**
    - Short Assignment: Honestly can be done by end of lecture, mainly running
  - Project presentations: **Dec 5th in class**
  - Final project reports: **Tuesday, December 12th**

- **Project presentations (5 min)**
  - TA's will send out sign up sheet

- **Final reports (make sure to include your name)**
  - 4-5 pages in conference paper format and style
  - Suggested outline on the course website.

Agenda

- **Stream Processing**
- **Recap**
- **Worktime: HW3 Snowflake + Project**

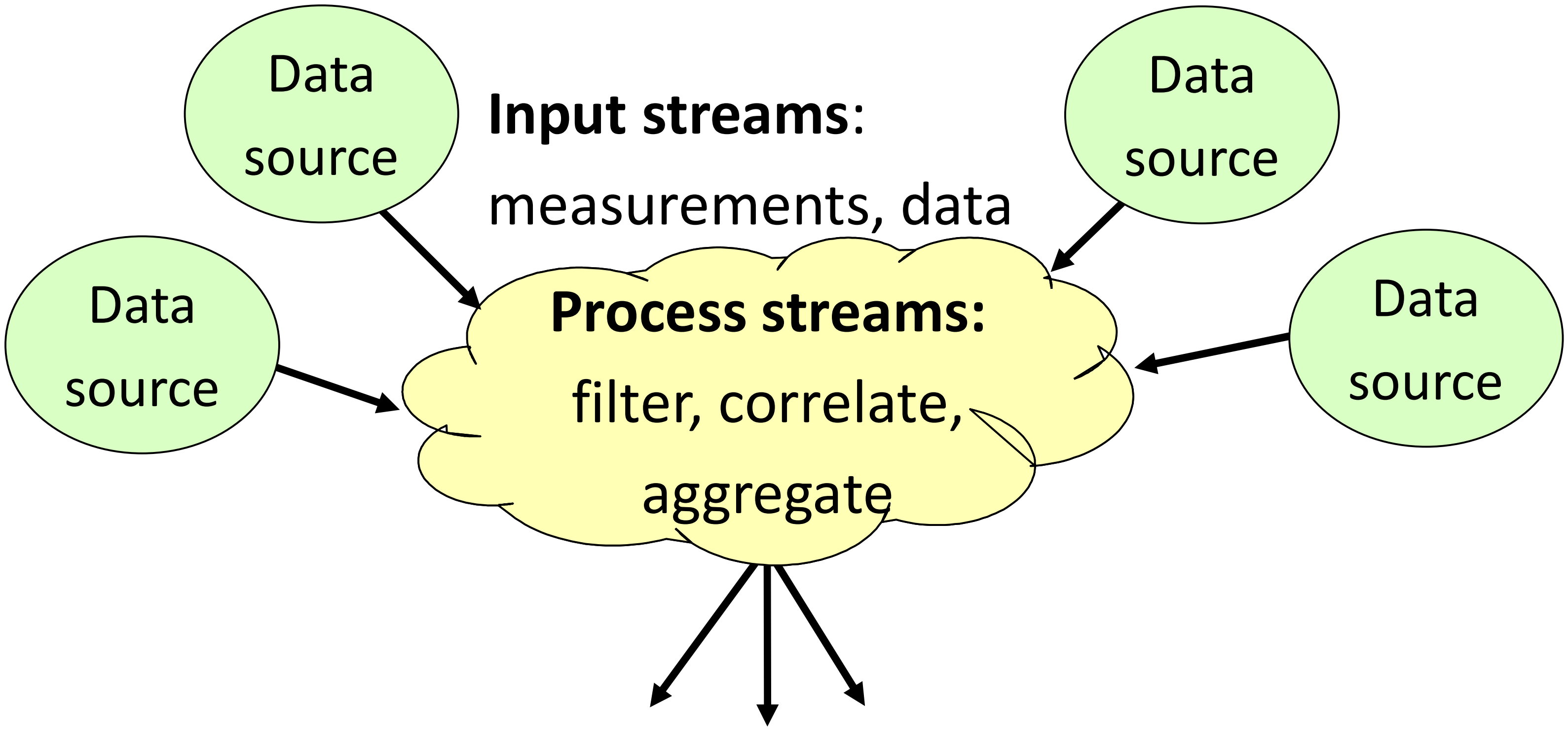
Stream Processing

Batch vs Stream Processing

- Batch Processing (Databases)
  - Data is present before queries are issued
  - Application rely on lots of stored information
  - Bounded Dataset
  - Finite Querying Time

- Stream Processing
  - Data is ingested and processed as it comes in
  - Applications rely on recent data and stored data
  - Unbounded Dataset
  - “Queries” can run for months/years/decades…

Stream Processing: Early Days



Output streams: alerts, anomalies, trends

Application Domains

- Network monitoring
  - Intrusion, fraud, anomaly detection, click streams
- Financial services
  - Market feed processing, ticker failure detection
- Sensor-based environment monitoring
  - Weather conditions, air quality, car traffic
  - Civil engineering, military applications, etc.
- Medical applications
  - Patient monitoring, equipment tracking
- Near real-time data analytics

Requirements

- **Input data is pushed continuously**
  – Traditional DBMSs not designed for continuous loading or inserting of individual data items
  – "DBMS-active, human passive" model

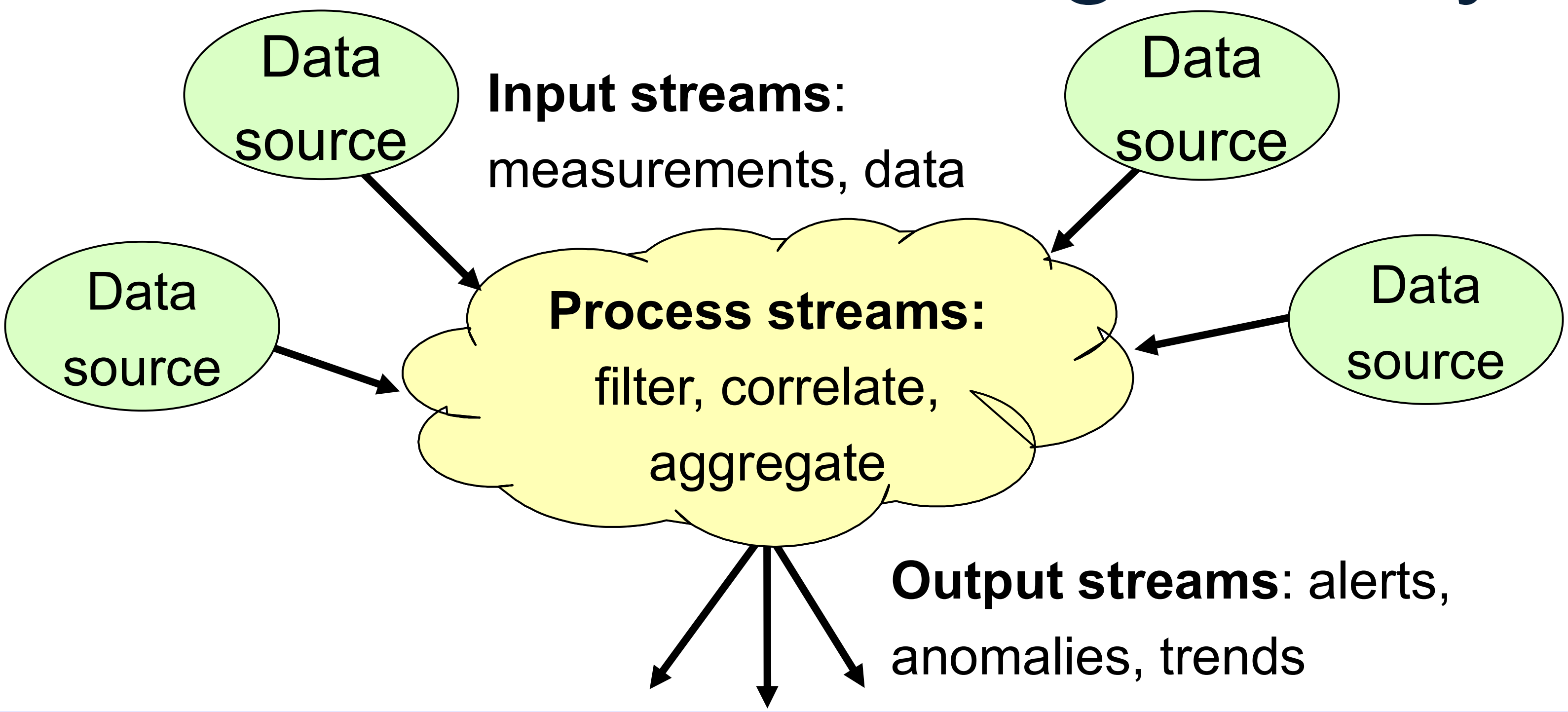
- **Users want to execute continuous queries**
  – Traditional DBMSs have no direct support for such queries. Can use triggers, but triggers do not scale

- **Low-latency processing**
  – Need to see results in near real-time
  – Data is possibly high-volume and high-rate

Other Requirements

- Distribution
- Load management and load shedding
- Approximate processing, approximate answers
- Fault-tolerance

Stream Processing: Today



Application domains: IoT, Web analytics, application telemetry, finance, healthcare

Stream processing engines: Kafka, Heron, Trill, StreamInsight, Spark Streaming, Beam, Flink, …

Same But Different Stream Processing

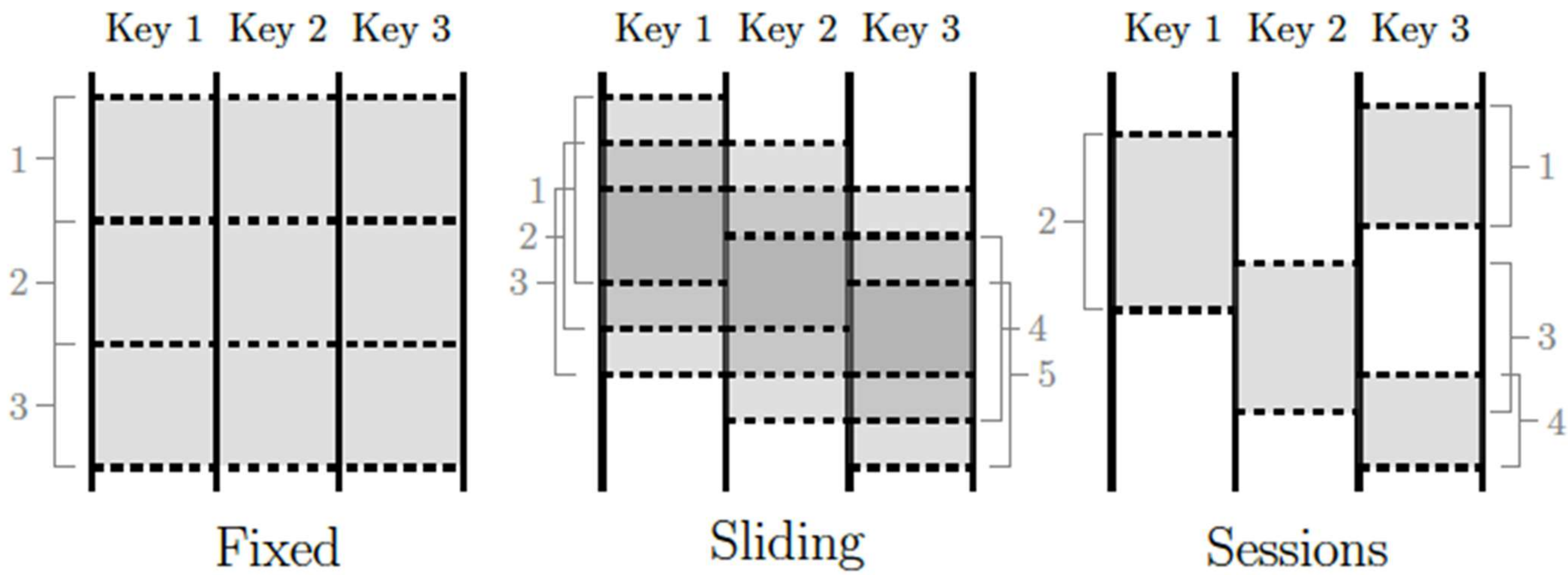
- Today, stream processing fundamentally same
  - Unbounded streams of tuples, timestamps, windows, …
- But recent systems have different emphases
  - Single programming model for batch and streaming
  - Parallel, shared-nothing stream processing
  - APIs in Python, Java, etc
  - Seamless support for user-defined functions

Streaming Concepts

Types of Windows

- Fixed (Tumbling) windows: Static window size
  - Hourly windows or daily windows
- Sliding windows: Overlapping static windows
  - Defined by a window size and a slide interval
  - Hourly window sliding by one min
- Sessions: Data-dependent windows
  - Group data by key
  - For each key, a window is a burst of data in the stream
  - Window ends when a timeout occurs

Types of Window Illustration

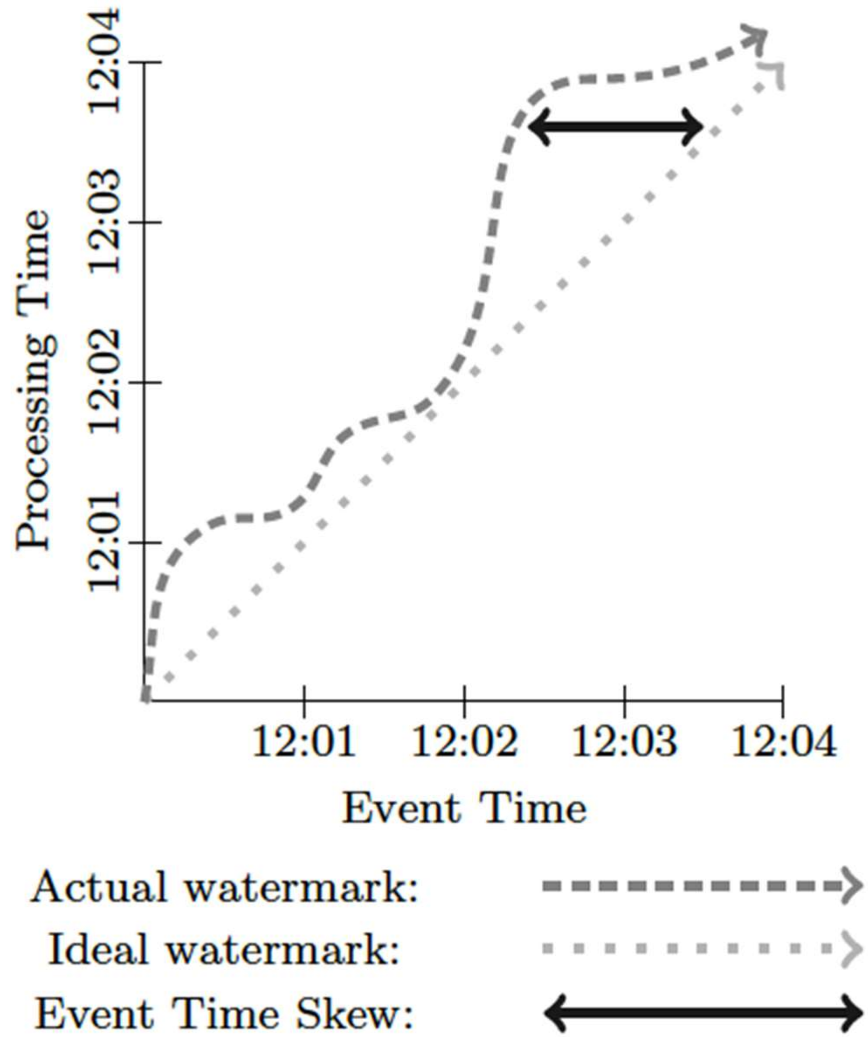


From: Akidau et. al. The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. VLDB'15.

Two Notions of Time in Streams

- Event time: Time when event occurred
- Processing time: Time when event observed
- Time domain skew
  - Difference between event and processing times
  - No skew: Process events immediately when they happen
- Watermark
  - Heuristic, lower bound on event time processed
  - Semantics: Future tuples should have higher timestamps
  - But, sometimes tuples can be late compared with watermark

Time Domain Skew Illustrated



From: Akidau et. al. The dataflow model:... VLDB'15.

Watermark Challenges

- Heuristic based
- If too fast, then data may be late
  - Tuple with event-time 9 min may arrive after watermark for event-time 10 min was emitted
- If too slow, may cause processing latencies
  - If we wait for a watermark before processing data such as in a window aggregation

Stream Processing Algorithms

Constraints

- Need to process elements as they arrive
- Can only use a small amount of memory
- No time to read from or write to disk

- Often, we will use *approximate* algorithms

General Sampling Approach

- To select a fraction a/b of stream elements
- One attribute in stream is key
  - The user ID in the next example
  - But it could be the search query or another attribute
- Hash key value into b buckets
- Retain all stream items that hash into first a buckets

Sampling from Streams

- Goal: Collect a representative sample of stream data
- Example:
  - Search engine receives a stream of queries
  - "What fraction of the typical user's queries were repeated over the past month?"
  - Wish to store only 1/10th of the stream elements
- Challenge:
  - If we pick 1/10th of all queries, hard to reason about complex user behavior such as fraction of repeated queries
  - Solution: pick 1/10th of users and keep all queries for those users
  - How? Hash user IDs into 10 buckets. If a user hashes into bucket 0, keep the corresponding query

Stream Selection

- Goal: Apply a filter to a stream
  - If a tuple meets the selection condition, keep it
  - Otherwise, drop it
- Challenge: Some filter predicates are expensive to compute
  - Example: Look up email address to decide if spam
- Solution: Bloom filters

Bloom Filters

0	0	0	0	0	0	0	0	0	0

S: Set of key values that pass the filter

For each value v in S, compute hash(v), set corresponding bit to 1

0	1	0	0	1	0	1	1	0	0

For each tuple in the stream with key value w

Compute hash(w)

If corresponding bit is 1, then tuple passes the filter

Bloom Filters

0	0	0	0	0	0	0	0	0	0

S: Set of key values that pass the filter

For each value v in S, compute hash(v), set corresponding bit to 1

0	1	0	0	1	0	1	1	0	0

For each tuple in the stream with key value w

Compute hash(w)

If corresponding bit is 1, then tuple passes the filter

Not Perfect?

Bloom Filters

0	0	0	0	0	0	0	0	0	0

S: Set of key values that pass the filter

For each value v in S, compute hash(v), set corresponding bit to 1

0	1	0	0	1	0	1	1	0	0

For each tuple in the stream with key value w

Compute hash(w)

If corresponding bit is 1, then tuple passes the filter

Improvement: Use K hash functions instead of one

For each key value, compute K hashes and check K bits

Counting Distinct Elements

- Goal: Compute the number of distinct values of an attribute in a stream

- Example: Count number of distinct visitors to website

- Challenge: What if too many distinct elements to hold in memory?

- Approach: Flajolet-Martin Algorithm

Flajolet-Martin Algorithm

- Tuple in stream with value w
- Compute hash(w) → $001010101011\overbrace{000}^{\text{Tail length}}$
- R is max tail length seen so far
- Estimate of distinct elements: 2^R
  – Divided by constant factor ~0.77351

- Extend to many hash functions
  – Take median of group averages

Materialize

Building with Real Data

- Sacrificing Speed
- Forgoing Features
- Compromising Cost

Materialized View

Traditional approaches to data:

1. Collect Data
2. Write to DB
3. Query the DB

Materialized View

Traditional approaches to data:

1. Collect Data
2. Write to DB
3. Query the DB

Expensive Operations at (3) Querying Time (Joins, Group By, etc) increases Latency

Materialized View

Materialized Views are precomputed query results whose output is stored for fast usage

In a streaming setting, these results need to be refreshed and updated to include recent data

Materialized View

Materialized Views are precomputed query results whose output is stored for fast usage

In a streaming setting, these results need to be refreshed and updated to include recent data

Streaming DBs allow definition of materialized views on incoming data and incrementally updates the views

Conclusion

- Stream processing was and still is an active research area
- It is now a key component of big data solutions in industry
- Many algorithms specialized for stream processing
- Today’s systems and techniques build on past work in database community

Course Review

Topics

- Relational Model
- Query Execution/Optimization
- MapReduce/Spark
- Parallel Query Evaluation
- Graphs
- Column Stores
- Streaming

Topics

- Relational Model
- Query Execution/Optimization
  - Relational Algebra
  - Data Independence
  - Logical/Physical Query Plans
  - Optimization
  - Cardinality Estimation

Topics

- MapReduce/Spark
- Parallel Query Evaluation
  - MapReduce
  - Spark
  - Parallel Query Plans
    - Hashing, Partitioning, Shuffling, etc

Topics

- Graphs
  - Recursions
  - SQL Limitations
- Column Stores
  - Design Choices
  - Efficiency
- Streaming