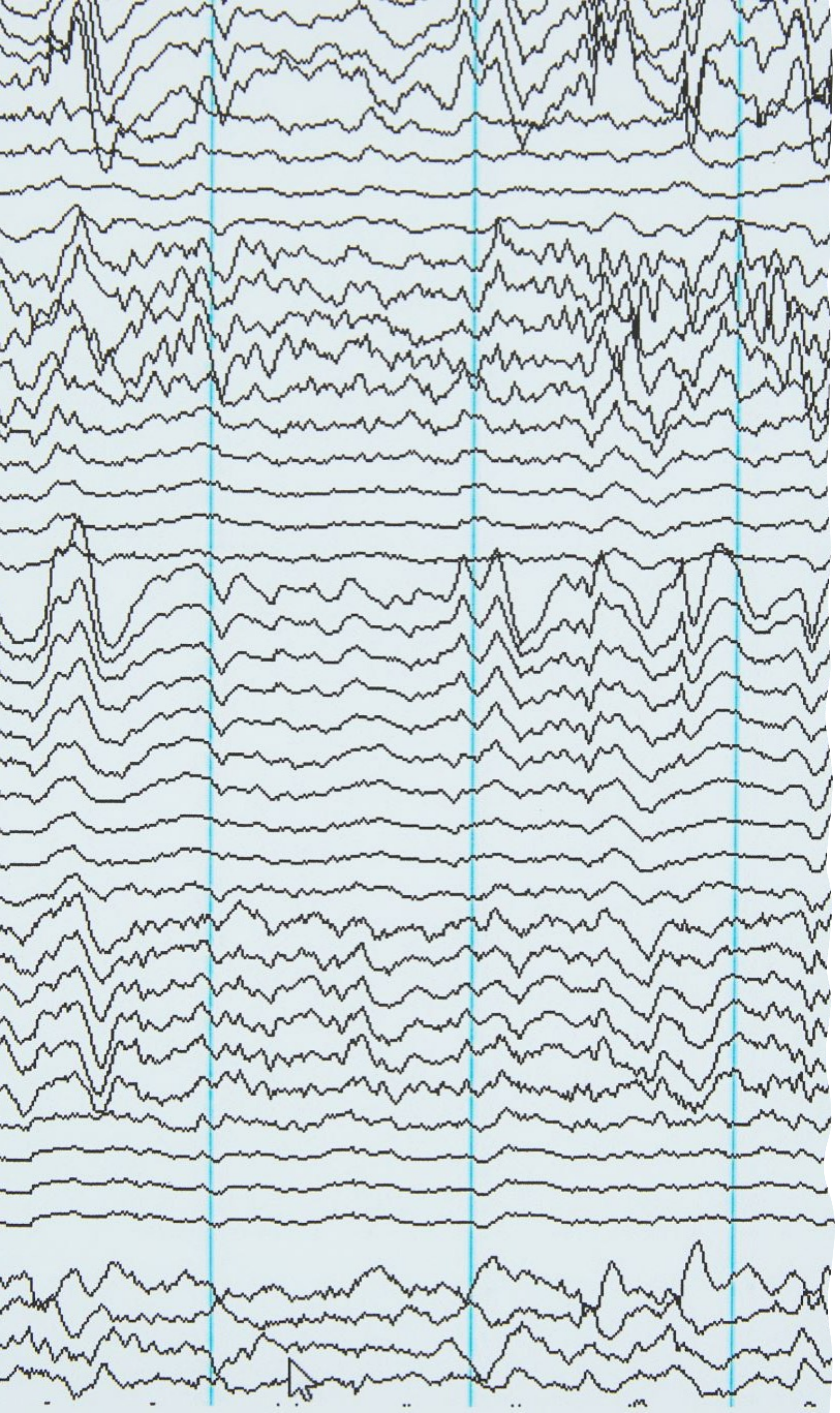
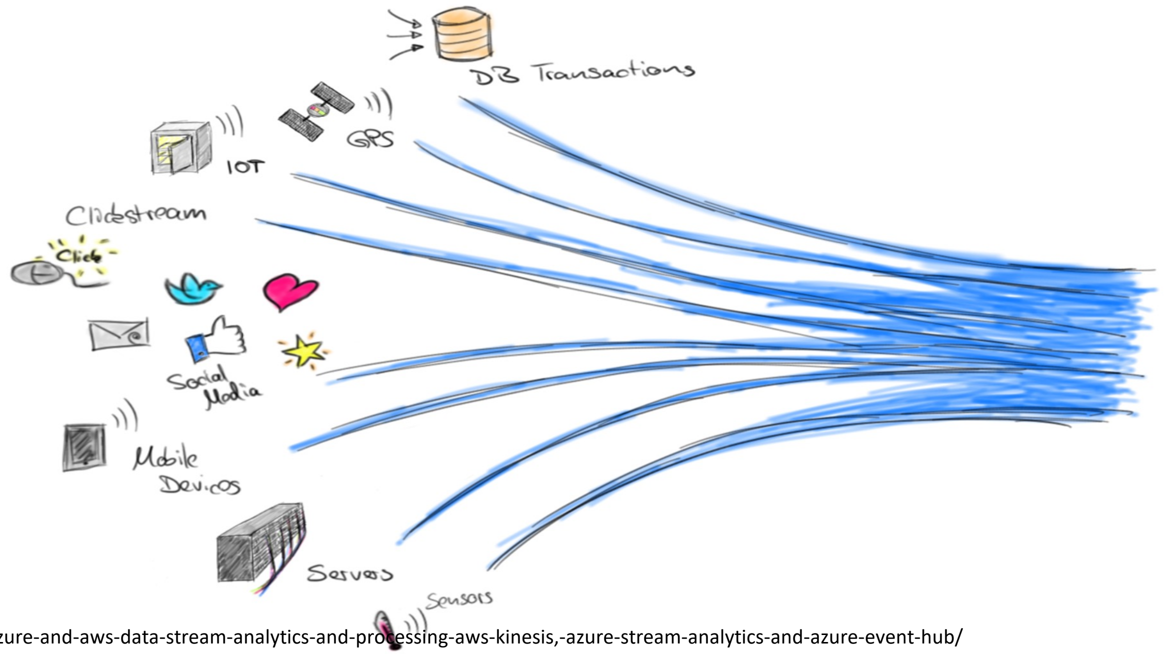


Data Streams

- Ravi Kumar Gupta
- https://kravigupta.in

Data Intensive Applications

- Data is modeled best as transient *data streams.*
- Not for persistent relational databases
- Traditional databases fail because of huge volume and velocity
- Can be thought of as infinite data
  - It never stops
  - We never know the entire set
- Examples
  - Financial applications
  - Network monitoring
  - Sensor networks
  - Blogging, twitter posts
  - Emails, Call records and more..

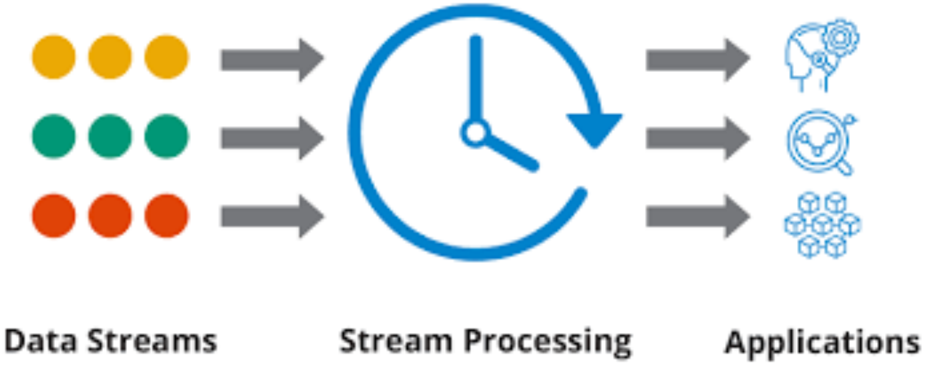


https://trojrobert.github.io/azure-and-aws-data-stream-analytics-and-processing-aws-kinesis,-azure-stream-analytics-and-azure-event-hub/

Data Streams

What is a data stream?

- A continuous, real-time flow of data
- Transient
- Time sensitive



- An individual data may be thought of as a relational database tuple
  - Sensor reading, tweet, web page visit, network measurement
- However, needs a way to handle because of
  - Rapid pace – Data arrives quickly and continuously
  - Time sensitive – Requires immediate action or analysis
  - Unpredictable – Data can arrive in varying volumes and at irregular intervals

Limitations of Traditional Databases

- Traditional Databases are designed for static, persistent data
- Not suitable for real-time analytics
- Can not handle continuous and rapid data flow
- Storage constraints for transient, high velocity data
- Rigid schema limits adaptability to changing data types
- Query latency not suitable for real-time decision making
- Difficulty in scaling horizontally
- Risk of data loss due to lack of real-time backup mechanisms

Importance of Immediate Processing

- Real-time decision making – Applications need quick responses
  - Imagine waiting a minute for responses to google queries
- Immediate processing of incoming transient data prevents loss of information
  - What if the recent tweets are not accounted for deciding trending topics?
- Efficient use of memory and CPU for analytics
- Timely insights and Adaptability – quick adjustments to algorithms and strategies
- Minimizes the risk of data corruption and loss
- Better user experience – enhance responsiveness in consumer facing applications
  - Imagine if you open Amazon/Flipkart/Netflix and you don’t have any recommendations

Challenges in Data Stream Mining

Algorithms

- must process the data with limited resources – time and memory
- Must deal with data whose nature or distribution changes over time

Data Stream Management Systems (DSMS)

Need of DSMS

- Real-Time Analytics: Enables immediate decision-making based on current data.
- Data Volume: Manages the high throughput of continuously arriving data.
- Scalability: Built to scale horizontally, accommodating increasing data loads.
- Query Support: Allows for continuous queries and real-time data manipulation.
- Resource Efficiency: Optimized for low-latency processing and minimal memory usage.
- Fault Tolerance: Provides mechanisms for data backup and recovery in real-time.
- Adaptability: Designed to handle changing data patterns and distributions.

Data Stream Model

- Data Stream – a real-time, continuous, and ordered(by time) sequence of items
- Not possible to control the order in which the items arrive
- Not feasible to store a stream entirely in any memory device

Querying Data

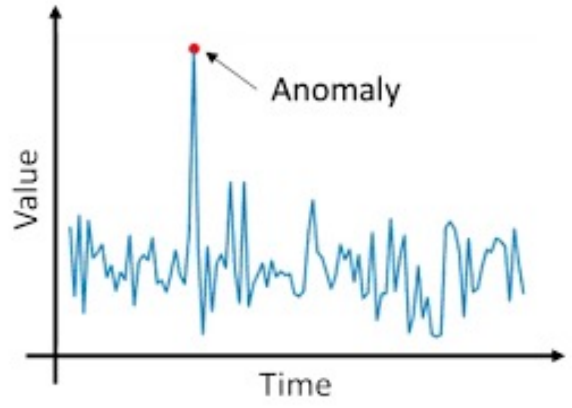
- If we query something, it will run continuously over a period of time
- Will return incremental new results as new data arrives
- Known as – long-running, continuous, standing, and persistent queries.

Data Stream Model: Characteristics

- Order & Time-Based Operations
  - Must support queries based on both sequence and time.
  - Queries like – Find the average temperature in last 10 minutes
- Approximate Summaries
  - All the data can not be stored, hence some approximate summary structure must be used
  - Queries over the summaries may not result exact answers
  - Example – Data sketches or histograms to provide an estimated count or averages
- Non-Blocking Operators
  - Query plans can't use operators that require full input first.
  - Such operators will block the query processor indefinitely
  - Example – Like SORT or GROUP By
  - Better to use sliding window to sort or group the most recent data points

Data Stream Model: Characteristics contd..

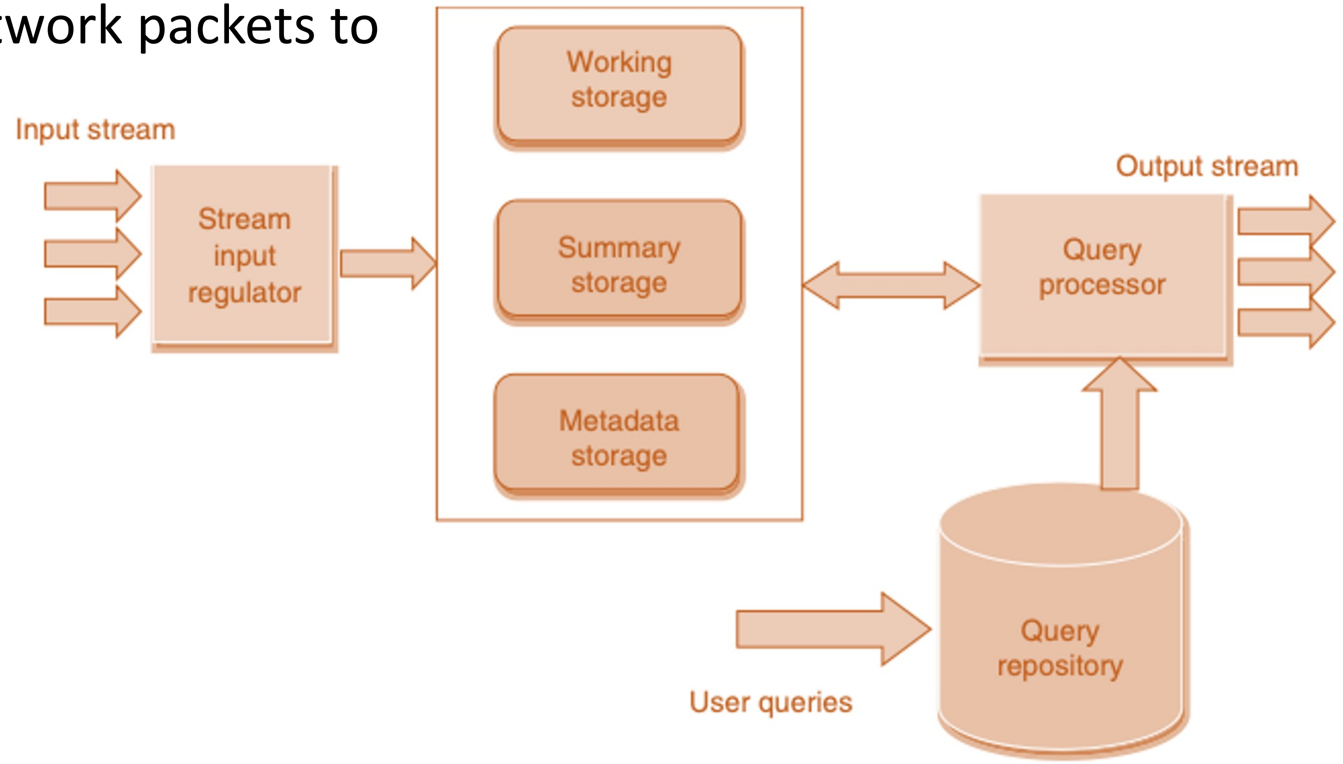
- Single-Pass Algorithms
  - Backtracking is infeasible due to storage and performance limits.
  - Algorithms needing only one pass of over the data would be better
- Real-Time Monitoring
  - Must adapt to changes and unusual data values quickly.
  - Anomaly detection algorithms which can flag any unusual data
  - Example – Spikes in network traffic in real-time
- Scalability
  - Must allow parallel and shared execution of continuous queries.
  - Distributing Query processing across multiple servers to handle large-scale data



DSMS Architecture

Input monitor

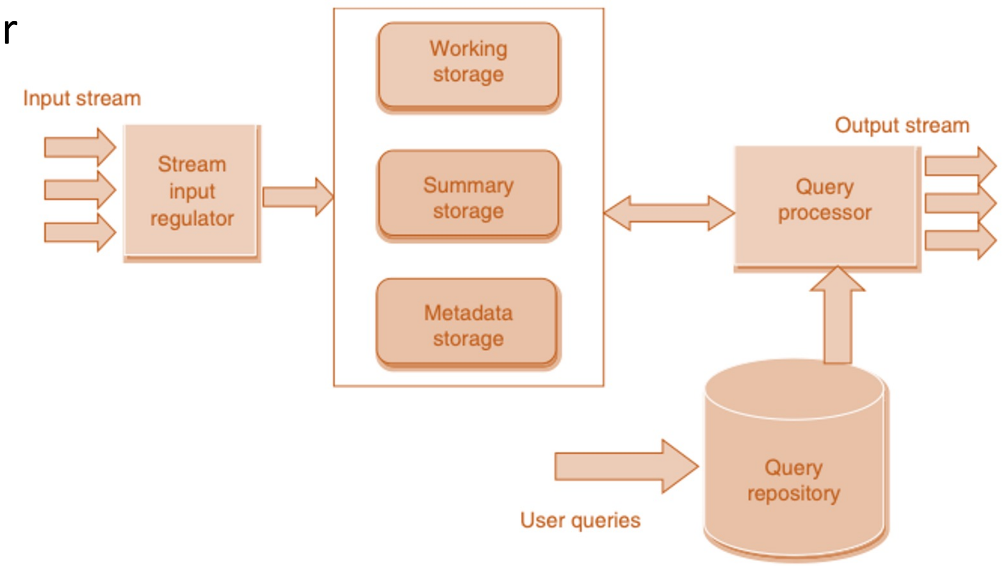
- Regulates input rates, may drop packets to manage flow.
- Example: Throttling incoming network packets to avoid system overload.



DSMS Architecture contd..

Data Storage Partitions

- Temporary Working Storage
  - For window queries and real-time analytics.
  - Example: Storing the last 10 minutes of sensor readings for quick retrieval.
- Summary Storage
  - Stores approximate summaries for efficient querying.
  - Example: Keeping a rolling average or data sketches for quick calculations.
- Static Storage for Meta-Data
  - Holds information like the physical location of each source.
  - Example: Storing IP addresses of IoT devices sending sensor data.



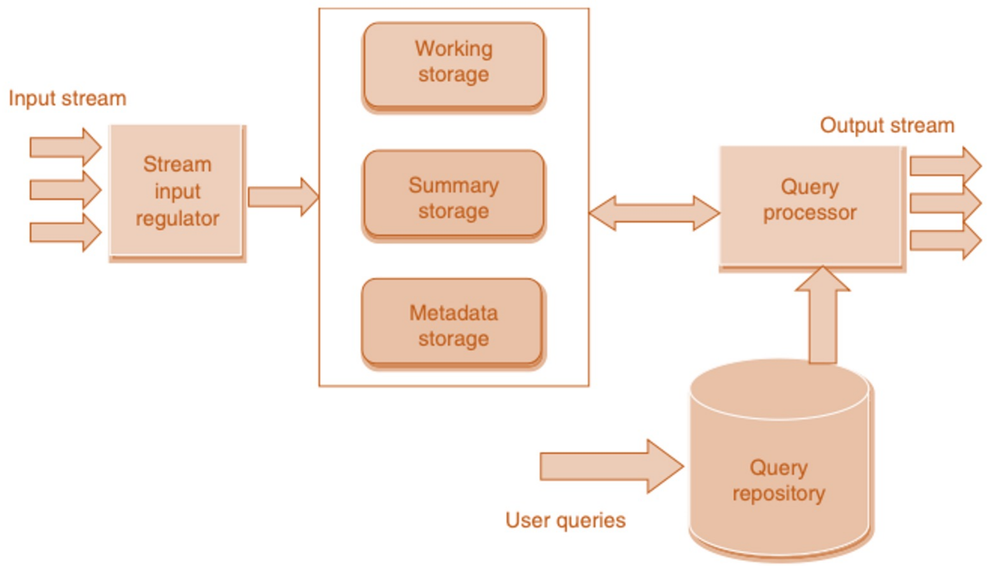
DSMS Architecture contd..

Query Repository

- Registers long-running queries and groups them for shared processing.
- Example: A query that continuously monitors for network intrusions.

Possibility of One-Time Queries

- Allows queries over the current state of the stream.
- Example: A query to fetch the current stock price.



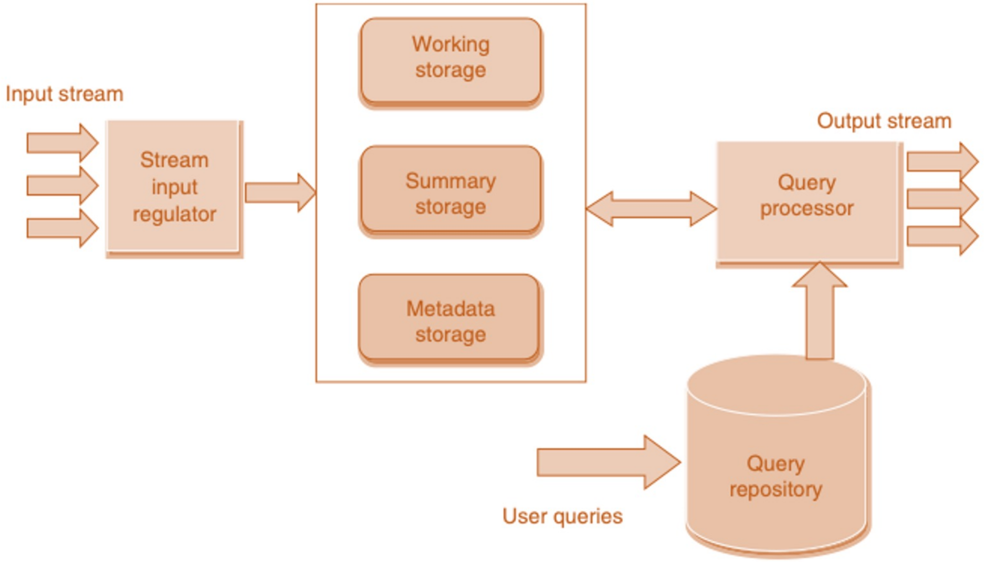
DSMS Architecture contd..

Query Processor

- Communicates with the input monitor.
- May re-optimize query plans due to changing input rates.
- Example: Adjusting query complexity if the data rate slows down.

Results Handling

- Streams results to users or temporarily buffers them.
- Example: Sending real-time analytics dashboards to users or storing results for later retrieval.



Data Stream Mining

Data Stream Mining

What is Data Stream Mining?

- Extracting knowledge from continuous, rapid data streams
  - Example: Real-time fraud detection in credit card transactions.
- Traditional algorithms – can not adapt to continuous data supply
  - Example: A machine learning model trained on past sales data can't instantly adapt to new sales trends without retraining.

Data Stream Mining contd..

Challenges

- **Volume**: High amount of data.
    - Social media platforms generating terabytes of user data daily.
    - Billions of tweets per day
    - IoT devices in smart cities generating massive amounts of sensor data.
- **Velocity**: Rapid data generation.
    - Stock market data that updates every millisecond.
    - Real-time monitoring of natural disasters like earthquakes or tsunamis.
- **Volatility**: Constantly changing data patterns.
    - Seasonal changes affecting energy consumption patterns.
    - News trends that can shift public opinion rapidly – affecting related metrics such as stock prices

Concept Drift

- **Concept Drift:** Changes in data patterns or behaviour over time.
  - Ex. A weather prediction model may need to adapt to climate change.
- Impact: Affects the accuracy and reliability of data mining models.
  - A spam filter may start failing if it doesn't adapt to new spamming techniques.
- Types
  - Sudden: Immediate change, e.g., viral trends.
  - Incremental: Gradual shift, e.g., changing consumer preferences.
  - Gradual: Alternating concepts, e.g., seasonal trends.
- Quick adaptation is crucial for real-time applications and maintaining data relevance
- Handled by adaptive algorithms, periodic re-training, and performance monitoring

Data Stream Mining contd..

Online Mining of Changes

- Importance: Real-time analysis for timely decisions.
  - Example: Immediate alerts for security breaches.

Technical Challenges

- Random Access: Difficult in fast, large streams.
  - Example: Can't access all tweets in real-time.
- Multi-Pass Algorithms: Infeasible due to volume and speed.
  - Example: Traditional clustering algorithms.

Core Assumptions

- Single Inspection: Data seen only once.
  - Example: Factory sensor data.
- Incremental Updates: Real-time model adaptation.
  - Example: Updating recommendation systems with each click.

Data Stream Applications

Sensor Networks

- Sensor Networks
- Network Traffic Analysis
- Financial Applications
- Transaction Log Analysis

Sensor Networks

Sensor networks generate massive, continuous streams of data for real-time monitoring and decision-making.

Use Cases

- Alerts and alarms based on sensor data.
- Aggregation and joins over multiple streams for complex analysis.

Representative Queries

- Disaster Alerts: Joining data streams like temperature and ocean currents to warn about natural disasters like cyclones and tsunamis.
  - Note: Information can change rapidly due to natural factors.
- Power Usage Monitoring: Continuously track power usage statistics, group by location or user type for efficient power distribution.

Network Traffic Analysis

Real-time analysis of network traffic for congestion management and security monitoring.

Use Cases

- Identifying and predicting network congestions.
- Detecting fraudulent activities like intrusion or denial of service attacks.

Situations can change drastically in a limited time, requiring immediate action.

Representative Queries

- Intrusion Detection: Compare current action streams over a time window to previously identified intrusions.
- Congestion Indicators: Analyse common intermediate nodes in traffic routes to identify potential congestion points.

Financial Applications

Real-time analysis of financial data for making timely investment decisions.

Use Cases

- Correlation identification, trend analysis, and to some extent, forecasting future stock valuations.

Data source - Constant inflow of data from news, current stock movements, and other market indicators.

Representative Queries

- Tax Cut Impact: Identify stocks priced between $50 and $200 with large buying in the last hour due to federal bank news about tax cuts.
- High-Performing Stocks: Find stocks trading above their 100-day moving average by more than 10% and with a trading volume exceeding a million shares.

Transaction Log Analysis

Real-time analysis of transaction logs for customer behavior insights and fraud detection.

Use Cases

- Web usage patterns, telephone call records, and ATM transactions.

Goal : Identify customer behaviour patterns and detect suspicious activities.

Representative Queries

- Customer Behaviour: Examine current buying patterns on a website to plan advertising campaigns and product recommendations.
- Fraud Detection: Continuously monitor credit card usage by location and average spending to identify potentially fraudulent activities

Stream Queries

Stream Queries

Queries over data streams share similarities with traditional DBMS but are adapted for continuous data.

Two types

1. One-time Queries
    - Evaluated once over a snapshot of the data set
    - Example: A stock price checker alerting when a stock crosses a specific price point.
2. Continuous Queries
    - Evaluated continuously as new data arrives.
    - Characteristics: Answers are produced over time, reflecting the data seen so far.
    - Storage: May be stored and updated or produced as new data streams.

Stream Queries contd..

- Aggregation Queries
  - Definition: Continuous queries for finding maximum, average, count, etc.
  - Example: Maximum stock price every hour.
  - Storage: Simple summaries like maximum or sum are stored, not the entire stream.

Stream Queries contd..

Join Queries

- Rapid:
  - Data streams can generate data at high rates.
  - Joining streams can produce a large number of results quickly.
  - Example: Joining streams of website clicks and purchases can quickly correlate user behavior.
- Unbounded:
  - Data streams are continuous and potentially infinite.
  - The number of results from join operations can be limitless.
  - As data keeps flowing, new results continuously emerge.
- Monotonic – Once a piece of data is processed and contributes to the result, the removal of that data from the stream won't affect the result.
  - Example – Counting the number of items in a stream is monotonic. Once an item is counted, removing it won't change the result
  - Finding the last item in the stream is not monotonic.

Stream Queries contd..

- Pre-defined Queries
  - Queries that are set up before any relevant data arrives.
  - Nature: Typically continuous, designed to monitor specific conditions or patterns.
  - Advantage - Optimized for performance since they're known in advance.
  - Example: A query set up to continuously monitor and alert when stock prices cross a certain threshold.
- Ad-Hoc Queries
  - Queries issued on-the-fly, after data streams have already begun.
  - Nature: Can be one-time or continuous, based on sudden insight or immediate needs
  - Challenges with Ad-Hoc Queries
    - Query Optimization: Difficult to optimize as they are not known in advance.
    - Data Referencing: May require data elements that have already arrived and potentially discarded.
  - Example: A sudden query to analyze traffic patterns after an unexpected event or incident.

Issues
in
Stream Query Processing

Issues in Data Stream Query Processing

- Unbounded Memory Requirements
- Approximate Query Answering
- Sliding Windows
- Batch Processing, Sampling and Synopses
- Blocking Operators

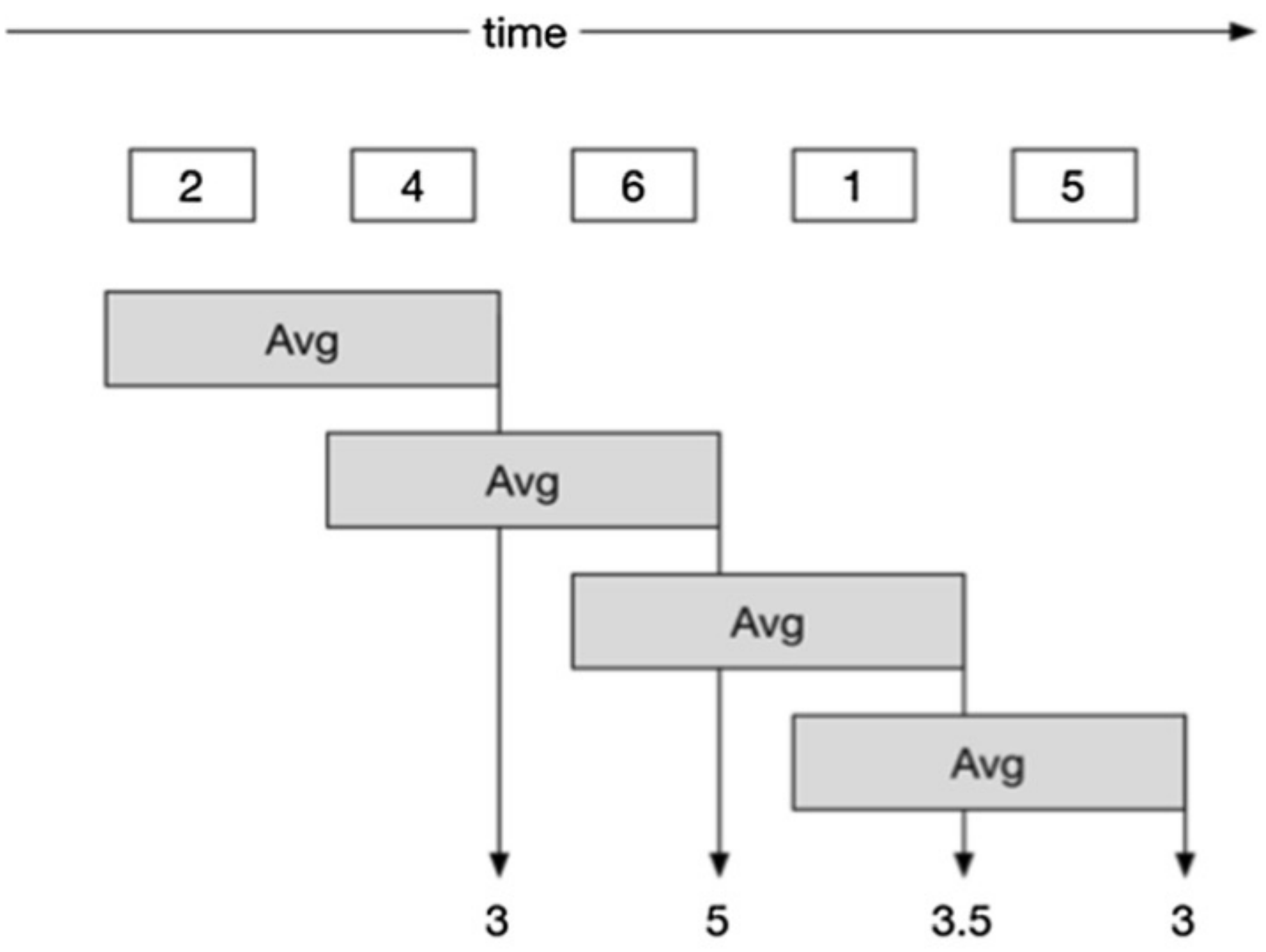
Unbounded Memory Requirements

- Data streams are potentially infinite, leading to unbounded memory requirements for exact query answers.
- Algorithms depending on external memory are not a good fit for stream processing
  - Slow, Do not support continuous queries
- Data never stops.
- New data constantly arrives while the old data is still in process
- High computation time can lead to increased latency – not good for real time, quick decisions
- Algorithms that operate within main memory without needing disk access are preferred.

Approximate Query Answering

- Challenge - Bounded memory makes exact answers challenging for data stream queries.
- Solution - High-quality approximate answers as an alternative to exact solutions.
- Approximation Algorithms - Growing area, focuses on data reduction and synopsis construction
- **Synopsis or Sketch** => Compact representation or summary of a larger dataset
- Key techniques –
  - **Sketches**: Compact representations of data.
  - **Random Sampling**: Using a subset of data to estimate properties of the entire dataset.
  - **Histograms**: Graphical representation showing data distribution.
  - **Wavelets**: Mathematical functions used to decompose data.
- Recent Developments:
  - Histogram-based techniques for correlated aggregate queries.
  - Small space summaries for various aggregate queries.

Sliding Windows



Sliding Windows

Sliding Windows

- Evaluate queries over recent data, not the entire history.
- Older data is discarded, keeping only the most recent data within the window.
- Benefits
  - Emphasis on Recent Data: Recent data is often more relevant and important in real-world applications.
  - Reduction in Data Volume: Limits the amount of data to be processed, making real-time analysis feasible.
- Example – Network traffic patterns, phone call or transaction records
- Types of sliding windows –
  - Count-based: Contains the most recent 'n' elements.
  - Time-based: Contains all elements from the last 't' time units (e.g., 1 month).

Batch Processing, Sampling And Synopsis

Batch Processing:

- Buffer data elements as they arrive and compute query answers periodically.
- Provides exact answers for a specific past moment, avoiding uncertainty about accuracy.
- **Bursty Streams**: Effective for data streams with bursts of data.
- **Trade-off**: Sacrifices real-time accuracy for timeliness.

Sampling:

- Skip some data points to evaluate queries over a sample, not the entire stream.
- **Reason**: Data often arrives faster than it can be processed.
- **Outcome**: Provides an approximate answer based on a representative subset of data.

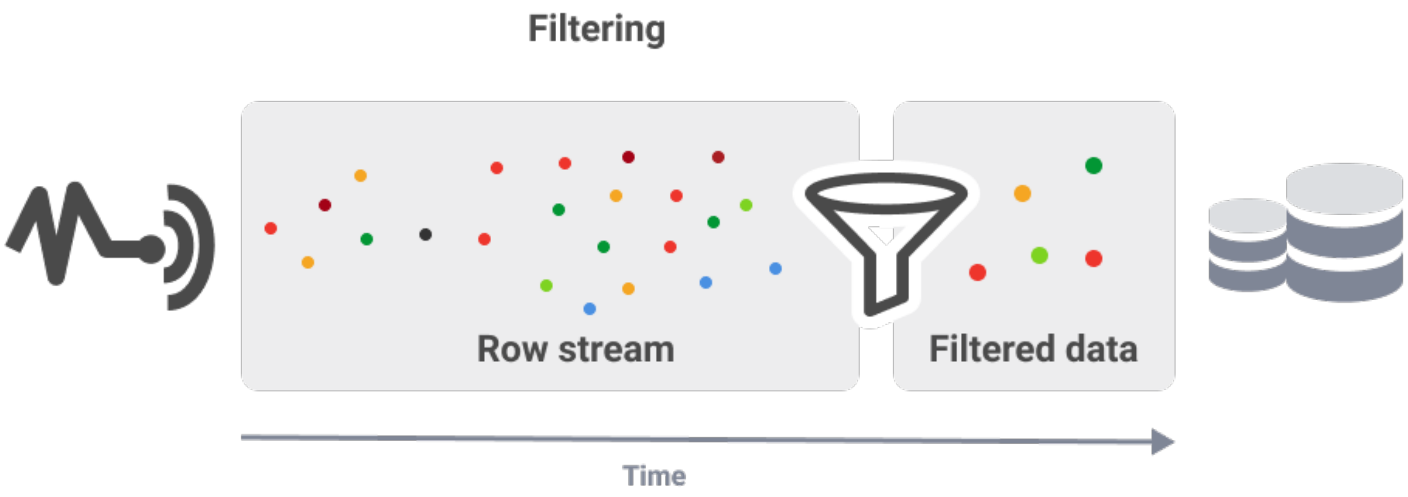
Synopsis or Sketch:

- **Definition**: A compact representation of data.
- **Benefit**: Reduces computation per data element, making processing more efficient.

Blocking Operators

- Operators that cannot produce results until they've seen their entire input.
- Examples – Sorting, Aggregation Operators: Such as SUM, COUNT, MIN, MAX, and AVG.
- Issues
  - Data streams can be infinite
  - If it does not see entire input, won’t produce output
- Not suitable for data stream computation model
- Challenge – Integrating blocking operators into data stream processing without compromising real-time results

Filtering Streams



Filtering Streams

- **Objective of Filtering**:
  - Reduce the initial volume of data.
  - Identify and retain items of interest for further evaluation.
- **Method**:
  - Continuously examine each item in the stream.
  - Decide if it should be stored based on predefined criteria.
  - Ex. Bloom Filter
- Bloom Filter
  - Used to test if an element belongs to a set

Filtering Streams

Example

- Context: Google Chrome's need to block dangerous URLs.
- Challenge:
  - Large number of malicious sites (~1 million).
  - URL length varies (2 to 2083 characters).
  - Storing all URLs in main memory isn't feasible.
  - High-velocity data stream due to continuous user access.

Solution – Bloom filter

Bloom Filter

Solution – Bloom filter

- Definition: A Bloom filter is a space-efficient probabilistic data structure used to test whether an element is a member of a set.
- Key Characteristics:
  - Probabilistic: Can tell you if an element is definitely not in the set or might be in the set.
  - Space-efficient: Uses much less memory than other data structures like hash tables.
- Memory Constraint: Only 1 megabyte of main memory available for the blacklist.
- Bloom Filter Mechanism:
  - Uses memory as a bit array (8 million bits).
  - A hash function maps each URL to one of the 8 million buckets.
  - The corresponding bit is set to 1 for each URL in the blacklist.

Bloom filter

- Initially all m bits of B are set to 0.
- Insert x into S. Compute $h_1(x), \ldots, h_k(x)$ and set

$$B[h_1(x)] = B[h_2(x)] = \ldots = B[h_k(x)] = 1$$

- Query if $x \in S$. Compute $h_1(x), \ldots, h_k(x)$.

If $B[h_1(x)] = B[h_2(x)] = \ldots = B[h_k(x)] = 1$*, then answer Yes, else answer No.*

Working of Bloom filter

- **Scenario**: Let's create a Bloom filter to check if a word is in a dictionary of three words: "apple", "banana", "cherry".
- **Bit Array**: Assume a bit array of size 10: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
- **Hash Functions**: Assume two simple hash functions, h1 and h2.
- **Insertion**:
  - "apple" hashes to positions 2 and 8. Set them to 1.
  - "banana" hashes to positions 4 and 7. Set them to 1.
  - "cherry" hashes to positions 3 and 8. Position 8 is already 1.
- **Final Bit Array**: [0, 0, 1, 0, 1, 0, 0, 1, 1, 0]
- **Query**:
  - For "apple", both positions 2 and 8 are 1. So, "apple" might be in the dictionary.
  - For "grape", assume it hashes to positions 5 and 9. Since position 5 is 0, "grape" is definitely not in the dictionary.

Bloom filter

Advantages:

- Highly space-efficient.
- Fast insertion and query operations.
- Suitable for applications where space is a constraint and a small probability of false positives is acceptable.

Limitations:

- Can't store the actual elements, only membership information.
- False positives are possible, but false negatives are not.
- Cannot remove elements from a basic Bloom filter.