Data Stream Management Systems
(An In-Memory System)

Monitoring Applications - Financial

Monitoring Applications - Traffic

Monitoring Applications - Environment

Employees at the control center of the Environmental Protection Monitoring Service in Beijing monitor pollution levels, 17 August 2007.
Monitoring Applications - Web

- **Travel Alerts** (TripCase and many apps)
  - Sends free updates to cell phone
    - Flight delays, gate changes, baggage claim, stormy weather

- **UV Alerts**
  - System integrates (1) customer’s location information with (2) ultraviolet radiation data from sensors
  - System alerts you to reapply sunscreen based on:
    - location, skin tone, and type of lotion!

Why are monitoring apps different?

- Data needs to be process continuously as it arrive
- Queries execute continuously
- Response time / latency in seconds (“real-time”)

Queries in Traditional DBMS

Continuous Queries in Traditional DBMS

- Store-then-query processing
- Store-then-query-and-query-again processing
Continuous Queries in Active DBMS

Circa 1990-2000

Store-store-then-trigger-processing

Data + Triggers

query

answer

Data + Triggers

Data Stream Management Systems

Circa 2001-2010

Data Stream Management Systems

Data Stream Management Systems

Continuous Queries

Data Stream Management Systems

Data Stream Management Systems

Answer Stream

DSMS = Conveyor Belt Model

AQSIOS

Our State of the art query processing system that efficiently utilizes resources

CPU time sharing:

Which operator to execute now?

And for how long?

What if the system is overloaded?

• Shed data to meet the near-real-time requirement

Which query plans are the best?

Scheduler

Load manager

Query optimizer

Continuous Queries

Stream applications

Data silos

Input queue

Query networks
Window Operators

- Window operator periodically produces sets of tuples
- Properties of *sliding windows*
  - **Range**: width of the window
  - **Slide**: How often to emit new sets of tuples (time, count)
- Aggregate CQ: Every 30 minutes, return the average stock price over the last hour
  - Range: 1 hour; Slide: 30 minutes

Aggregate Continuous Queries (ACQs)

- SELECT <Aggregation(attribute)>
- FROM <named_stream_list>
- WHERE <Predicate(attribute-list)>
- GROUP BY <attribute list>
- **Range** = <user preference> and **Slide** = <user preference>

- SELECT AVG(trade) FROM stock_stream
  - Range = 1 hour and Slide = 10 mins
- SELECT AVG(trade) FROM stock_stream
  - Range = 8 hours and Slide = 1 hour

How to optimize the processing of multiple ACQs?

Processing Aggregate Continuous Queries

- **Naïve Implementation**: After each slide-interval: Produce the aggregate result of a new window instance from scratch (i.e., raw tuples)
- **Partial Aggregation**: [Li et al., SIGMOD'05]
- **Paired Windows**: [Krishnamurthy et al., SIGMOD'06]

Paired Windows

A slide is composed of 2 fragments:
- \( g_1 = \text{range} \% \text{slide} \) (\( g_1 = 8 \% 5 = 3 \))
- \( g_2 = \text{slide} - g_1 \) (\( g_2 = 5 - 3 = 2 \))

- \( \text{SUM}(g_1, g_2, g_3) \)
- Final result
- Partial results
- Input tuples
- Final-aggregate \( F() \)
- Sub-aggregate \( S() \)
Continuous Query Scheduling

Wait a minute!

There is OS scheduling. Why is this not enough?

Round-Robin Scheduling (Operators)

- Each operator is considered to be a task

Round-Robin Scheduling (Queries)

Two-level scheduling:
- Queries execute in round-robin
- Operators within queries are executed in pipelined fashion

Continuous Queries (CQ)

Example: Alert when IBM stock goes above $125

- CQ = a set of operators
Continuous Queries (CQ)

Example: Alert when IBM stock goes above $125
- CQ = a set of operators

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Symbol</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:01</td>
<td>MSFT</td>
<td>$28</td>
</tr>
<tr>
<td>10:02</td>
<td>IBM</td>
<td>$90</td>
</tr>
<tr>
<td>10:07</td>
<td>IBM</td>
<td>$127</td>
</tr>
</tbody>
</table>

CQs

Example: Alert when IBM stock goes above $125
- CQ = a set of operators

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Symbol</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:01</td>
<td>MSFT</td>
<td>$28</td>
</tr>
<tr>
<td>10:02</td>
<td>IBM</td>
<td>$90</td>
</tr>
<tr>
<td>10:07</td>
<td>IBM</td>
<td>$127</td>
</tr>
</tbody>
</table>

Each operator is associated with 2 properties:
- Operator Cost (c)
  - Processing time
- Operator Selectivity (s)
  - Not all inputs are events!
Event-based QoS Metric: Response Time

- The interval between an event’s arrival and departure
  - Example: Time between IBM=$125 and notification
  - Metric: Response Time (RT), Delay, Latency, ...
  - Discarded tuples do not contribute to the metric

- Average Response Time for N events = \( \frac{1}{N} \sum_{i=1}^{N} RT_i \)

- Response Time = processing time + queuing delay

Example: Impact of Selectivity

<table>
<thead>
<tr>
<th>Selectivity</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>1.0</td>
</tr>
<tr>
<td>Q2</td>
<td>0.2</td>
</tr>
</tbody>
</table>

- Input: 5 pending tuples arrived at time 0

Highest Rate Policy

- “Shortest time to detect an event” = “highest output rate”
- Assign each query a priority equals to its output rate
  \[ OR_i = \frac{S_i}{C_i^{avg}} \]
- At each scheduling point:
  schedule the query with the highest global output rate…

Highest Rate Policy (HR)

[Sharaf et. al., VLDB’06, ACM TODS08]

Summary

- **DBMS** = Database Management System
  - Data are stored
  - Queries come in
  - Answers come out, based on current state of database
  - Useful when updates come in batches and are not many

- **DSMS** = Data Stream Management System
  - Queries are stored (registered, ahead of time)
  - Data come in continuously, as a stream
  - Answers come out continuously, as a stream
  - Useful for monitoring applications
Continuous / Real-time Analytics

circa 2011- now

(1) Collect real-time data.  (2) Process data as it flows.  (3) Explore and visualize.

DSMS Challenges

- Given:
  - Continuous data streams [Variety]
  - High volume of updates [Volume]
  - Bursty data arrival [Variability]
  - Near real-time processing [Velocity]
  - Unreliable sources & connections [Veracity]
  - Protect privacy & provide security [Visibility]
  - Explore & visualize [Value]

  How to efficiently handle this .... Big Data?

How to handle the DSMS Challenges

- Query Optimization
  - new architectures, distribution/Cloud
  - New operator implementations
- Load Balancing
  - scale-out, query & operator migration,
  - shedding (drop tuples to meet response time)
- Scheduling of Continuous Queries
  - QoS
- Privacy and Security
- What about a "Store" operator?
  - consistency, reliability/fault tolerance

AstrapiDB: Real-time Analytics