Data Stream Systems

Reynold Cheng
12th July, 2002

Based on slides by B. Babcock et al., "Models and Issues in Data Stream Systems", PODS'02.

Outline of this Talk

- An Overview of Streams
- Data and Query Models
- Approximation Queries
- Other Research Issues
Data Streams

- Traditional DBMS – data stored in finite, persistent data sets
- New Applications – data input as continuous, ordered data streams
- A data stream as a growing relational table of potentially infinite size

Using Traditional Database
New Approach for Data Streams

User/Application

Register Query

Stream Query Processor

Results

Scratch Space
(Memory and/or Disk)

Data Stream Management System (DSMS)
Sample Applications

- Network management and traffic engineering (e.g., Sprint)
  - Streams of measurements and packet traces
  - Queries: detect anomalies, adjust routing
- Telecom call data (e.g., AT&T)
  - Streams of call records
  - Queries: fraud, customer call patterns, billing

Sample Applications

- Sensor Networks
  - Large number of cheap, wireless sensors
  - Streams of real-world measurements
  - Queries: monitoring, aggregate, alert
- Web tracking and personalization (e.g., Yahoo, Google)
  - Clickstreams, user query streams, log records
  - Queries: monitoring, analysis, personalization
Challenges

- Multiple, continuous, rapid, time-varying, ordered streams
- Main memory computations
- Queries may be continuous (not just one-time)
  - Evaluated continuously as stream data arrives
  - Answer updated over time
- Queries may be ad-hoc
- Beyond relational queries (scientific, data mining)

Meta-Questions

- Killer-apps
  - Application stream rates exceed DBMS capacity?
  - Can DSMS handle high rates anyway?
- Motivation
  - Need for general-purpose DSMS?
  - Not ad-hoc, application-specific systems?
- Non-Trivial
  - DSMS = merely DBMS with enhanced support for triggers, temporal constructs, data rate mgmt?
DBMS versus DSMS

- Persistent relations
- Transient streams

- Persistent relations
- One-time queries
- Transient streams
- Continuous queries
### DBMS versus DSMS

<table>
<thead>
<tr>
<th>DBMS</th>
<th>DSMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persistent relations</td>
<td>Transient streams</td>
</tr>
<tr>
<td>One-time queries</td>
<td>Continuous queries</td>
</tr>
<tr>
<td>Random access</td>
<td>Sequential access</td>
</tr>
</tbody>
</table>

- Persistent relations
- One-time queries
- Random access
- “Unbounded” disk store

- Transient streams
- Continuous queries
- Sequential access
- Bounded main memory
DBMS versus DSMS

- Persistent relations
- One-time queries
- Random access
- “Unbounded” disk store
- Only current state matters
- Relatively low update rate

- Transient streams
- Continuous queries
- Sequential access
- Bounded main memory
- History/arrival-order is critical
- Possibly multi-GB arrival rate
DBMS versus DSMS

- Persistent relations
- One-time queries
- Random access
- “Unbounded” disk store
- Only current state matters
- Relatively low update rate
- No real-time services

- Transient streams
- Continuous queries
- Sequential access
- Bounded main memory
- History/arrival-order is critical
- Possibly multi-GB arrival rate
- Real-time requirements

- Transient streams
- Continuous queries
- Sequential access
- Bounded main memory
- History/arrival-order is critical
- Possibly multi-GB arrival rate
- Real-time requirements
- Data stale/imprecise
Outline of this Talk

- An Overview of Streams
- Data and Query Models
- Approximation Queries
- Other Research Issues
Data Model

- Append-only
  - Call records
- Updates
  - Stock tickers
- Deletes
  - Transactional data
- Meta-Data
  - Control signals, punctuations

System Internals – probably need all above

Query Model

Query Registration
- Predefined
- Ad-hoc
- Predefined, inactive until invoked

Answer Availability
- One-time
- Event/timer based
- Multiple-time, periodic
- Continuous (stored or streamed)

Stream Access
- Arbitrary
- Weighted history
- Sliding window

DSMS
Example Queries

Query 1 (self-join)

- Find all outgoing calls longer than 2 minutes

SELECT   O1.call_ID, O1.caller
FROM      Outgoing O1, Outgoing O2
WHERE      (O2.time - O1.time > 2
            AND   O1.call_ID = O2.call_ID
            AND   O1.event = start
            AND   O2.event = end)

- Result requires unbounded storage
- Can provide result as data stream
- Can output after 2 min, without seeing end
Query 2 (join)

• Pair up callers and callees

  SELECT O.caller, I.callee
  FROM Outgoing O, Incoming I
  WHERE O.call_ID = I.call_ID

• Can still provide result as data stream

• Requires unbounded temporary storage

Query 3 (group-by aggregation)

• Total connection time for each caller

  SELECT O1.caller, sum(O2.time – O1.time)
  FROM Outgoing O1, Outgoing O2
  WHERE (O1.call_ID = O2.call_ID
      AND O1.event = start
      AND O2.event = end)
  GROUP BY O1.caller

• Cannot provide result in (append-only) stream
  • Output updates?
  • Provide current value on demand?
Outline of this Talk

- An Overview of Streams
- Data and Query Model
- **Approximation Queries**
- Other Research Issues

Impact of Limited Memory

- Continuous streams grow unboundedly
- Queries may require unbounded memory
- [ABBMW 02]
  - a priori memory bounds for query
  - Conjunctive queries with arithmetic comparisons
  - Impact of duplication elimination
- Open – general queries
Approximate Query Evaluation

- **Why?**
  - Handling load – streams coming too fast
  - Data stream is archived in a off-site data warehouse, expensive access of archived data
  - Avoid unbounded storage and computation
  - Ad hoc queries need approximate history
  - Try to look at the data items only once and in a fixed order

Approximate Query Evaluation

- **How?** Sliding windows, synopsis, samples, load-shed
- **Major Issues?**
  - Metric for set-valued queries
  - Composition of approximate operators
  - How is it understood/controlled by user?
  - Integrate into query language
  - Query planning and interaction with resource allocation
  - Accuracy-efficiency-storage tradeoff and global metric
Synopses

- Queries may access or aggregate past data
- Need bounded-memory history-approximation
- Synopsis?
  - Succinct summary of old stream tuples
  - Like indexes/materialized-views, but base data is unavailable
- Examples
  - Sliding Windows
  - Samples
  - Sketches
  - Histograms
  - Wavelet representation

Sketching Techniques

- Self-Join Size Estimation
- Stream of values from $D = \{1, 2, \ldots, n\}$
- Let $f_i =$ frequency of value $i$
- Consider $S = \sum f_i^2$, or Gini’s index of homogeneity.
- Useful in parallel DB applications, error estimation in query result size estimation and access plan costs.
- Equivalent query: count ($R \ |><|_D R$)
Evaluating $S = \sum f_i^2$

- To update $S$, keep a counter $f_i$ for each value $i$ in the domain $D \Rightarrow \Omega(n)$ space
- Has to be kept for each self-join
- Question – estimating $S$ in sub-linear space? ($O(\log n)$)

Self-Join Size Estimation

- AMS Technique (randomized sketches)
  - Given $(f_1, f_2, \ldots, f_N)$
  - $Z_i = \text{random}\{-1, 1\}$
  - $X = \sum f_i Z_i$ (X incrementally computable)
- Theorem $\text{Exp}[X^2] = \sum f_i^2$
  - Cross-terms $f_i Z_i f_j Z_j$ have 0 expectation
  - Square-terms $f_i Z_i f_i Z_i = f_i^2$
- Space $= \log (N + \sum f_i)$
- Independent samples $X_k$ reduce variance
Estimation Quality

- How can independent samples $X_k$ improve the quality of estimation?
- Keep $s_1 \times s_2$ samples for $X_k$
- $s_1$ reduces variance, $s_2$ boosts confidence

Sample Run of AMS

$$V = \begin{bmatrix} 3 & 6 & 2 & 5 & 7 \end{bmatrix}$$

$$Z_1 = \begin{bmatrix} 1 & 1 & -1 & 1 & -1 \end{bmatrix} \quad Z_2 = \begin{bmatrix} -1 & 1 & -1 & 1 & 1 \end{bmatrix}$$

$$\sum v_i^2 = 123 \quad X_1 = 5, X_1^2 = 25 \quad X_2 = 14, X_2^2 = 196 \quad \text{Est} = 110.5$$

$$V = \begin{bmatrix} 4 & 6 & 2 & 5 & 7 \end{bmatrix}$$

$$Z_1 = \begin{bmatrix} 1 & 1 & -1 & 1 & -1 \end{bmatrix} \quad Z_2 = \begin{bmatrix} -1 & 1 & -1 & 1 & 1 \end{bmatrix}$$

$$\sum v_i^2 = 130, \quad X_1 = 6, X_1^2 = 36, \quad X_2 = 12, X_2^2 = 144, \quad \text{Est} = 90$$
Comments on AMS

- The self-join size can be computed on-line
- Sufficiently small variance (controlled by $s_1$ and $s_2$)
- Can this method be extended to answer other queries?

Complex Aggregate Queries

- A. Dobra et al. extend the idea of AMS to provide approximate answers to complex aggregate queries.
- SELECT AGG FROM $R_1, R_2, …, R_r$ where $E$
- AGG: COUNT/SUM/AVERAGE
- $E$: conjunction of $(R_i.A_j = R_k.A_l)$
- It is proved that the error of these estimates is at most $\varepsilon$ with probability $1-\delta$. 
Basic Notions of Approximation

- For aggregate queries (e.g., SUM, COUNT), approximation quality can be measured by relative error:
  - \( \frac{(\text{Estimated value} - \text{Actual value})}{\text{Actual value}} \)
- Open question: for queries involving more than simple aggregation, how should we define approximation?
- Consider \( S |<|_{B} T: (S: \{A,B\}, T: \{B,C\}) \)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20.5</td>
<td>Doctor</td>
<td>8</td>
<td>10.3</td>
<td>Lawyer</td>
</tr>
<tr>
<td>8</td>
<td>10.3</td>
<td>Lawyer</td>
<td>3</td>
<td>10.2</td>
<td>Teacher</td>
</tr>
</tbody>
</table>

Actual Result | Approximate Result

Basic Notions of Approximation (2)

- Can we accept this kind of approximation?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20.5</td>
<td>Doctor</td>
<td>11</td>
<td>21.6</td>
<td>Doctor</td>
</tr>
<tr>
<td>8</td>
<td>10.3</td>
<td>Lawyer</td>
<td>8</td>
<td>10.3</td>
<td>Student</td>
</tr>
<tr>
<td>3</td>
<td>10.2</td>
<td>Teacher</td>
<td>3</td>
<td>10.2</td>
<td>Teacher</td>
</tr>
</tbody>
</table>

Actual Result | Approximate Result
Basic Notions of Approximation (3)

- Can we provide useful (semantically correct) but stale results?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20.5</td>
<td>Doctor</td>
</tr>
<tr>
<td>8</td>
<td>10.3</td>
<td>Lawyer</td>
</tr>
<tr>
<td>3</td>
<td>10.2</td>
<td>Teacher</td>
</tr>
</tbody>
</table>

Table: Actual Result (at time t)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20.5</td>
<td>Doctor</td>
</tr>
<tr>
<td>8</td>
<td>10.3</td>
<td>Lawyer</td>
</tr>
</tbody>
</table>

Table: Approximate Result (correct result at time t - 6)

Outline of this Talk

- An Overview of Streams
- Data and Query Model
- Approximation Queries
- Other Research Issues
Data Mining

- High-Speed Stream Data Mining
  - Association Rules
  - Stream Clustering
  - Decision Trees
- **Single-pass** algorithms for inferring interesting patterns *on-line* (as the data stream arrives)
- Useful for mission-critical tasks like telecom fraud detection

Conclusion: Future Work

- Query Processing
  - Stream Algebra and Query Languages
  - Approximations
  - Blocking Operators, Constraints, Punctuations
- Runtime Management
  - Scheduling, Memory Management, Rate Management
  - Query Optimization (Adaptive, Multi-Query, Ad-hoc)
  - Distributed processing
- Synopses and Algorithmic Problems
- Systems
  - UI, statistics, crash recovery and transaction management
  - System development and deployment
References


A. Arasu, B. Babcock, S. Babu, J. McAlister, J. Widom. *Characterizing Memory Requirements for Queries over Continuous Data Streams, PODS ‘02.*

A. Dobra, M. Garofalakis, J. Gehrke, R. Rastogi. *Processing Complex Aggregate Queries over Data Streams, SIGMOD ‘02.*


Thank You!