Monitoring Streams: A New Class of Data Management Applications

CPSC 504: DATA MANAGEMENT 2009
PRESENTER: YONG
DISCUSSION: BRENDAN

Outline

- Motivation
  - 5 assumptions of traditional DBMS
  - Monitoring applications
  - Rethink the fundamental
- Aurora System Model
- Aurora Run-time architecture
  - QoS in Aurora
  - Real-time Scheduling
- Conclusion

5 assumptions of traditional DBMS

1. Passive repository: Human-Active, DBMS-Passive (HADP) model
2. The current state of the data is important: Previous data needs to be extracted from the log
3. Triggers and alerts as second-class citizens
4. Perfect synchronization of data elements and exact query answers
5. No real-time services from applications

So what’s wrong with this assumption?

Monitor application inside your car

So what’s wrong with this assumption?

<table>
<thead>
<tr>
<th>Monitoring Application</th>
<th>Traditional DBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical model</td>
<td>Data Active</td>
</tr>
<tr>
<td></td>
<td>Human Passive</td>
</tr>
<tr>
<td></td>
<td>Data Passive</td>
</tr>
<tr>
<td></td>
<td>Human Active</td>
</tr>
<tr>
<td>Managing History of values</td>
<td>required</td>
</tr>
<tr>
<td></td>
<td>Very hard or inefficient</td>
</tr>
<tr>
<td>Approximate query result</td>
<td>required</td>
</tr>
<tr>
<td></td>
<td>Not supported</td>
</tr>
<tr>
<td>Trigger oriented</td>
<td>required</td>
</tr>
<tr>
<td></td>
<td>Limited support</td>
</tr>
<tr>
<td>Real-time requirement</td>
<td>required</td>
</tr>
<tr>
<td></td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Market Analysis
- Forecast of stock exchange data
- Critical care
- Streams of vital signs measurements
- Physical plant monitoring
- Streams of environmental readings
- Biological population tracking
- Streams of positions from individuals of a species

5 assumptions

1. HADP model
2. Only the current data is important
3. Triggers and alerts as second-class citizens
4. Perfect synchronization of data elements and complete data
5. No real-time services
So what’s wrong with this assumption?

SO!

All 5 assumptions are problematic for motoring applications!

Aurora System Model

- So, the solution “Aurora”, which is designed to better support monitoring applications
  - Stream data
  - Triggers
  - Imprecise data
  - Real-time requirement

Aurora System Model

Aurora: process incoming streams in the way defined by an application (data-flow system: Aurora Network)

Data sources (stream): A stream in Aurora is a sequence of tuples from a given data source, and each tuple is time stamped upon entry to Aurora

Boxes: performs operations on incoming stream of data

Aurora Run-time architecture

3 kinds of query supported
  - Continuous
  - View
  - Ad-Hoc Query

Boxes: Operations

8 primitive operators (Box)

- Windowed: Operate on a set of consecutive tuples from a stream at a time. Applies function to a window and advances the window to capture a new set of tuples.
  - Slide: advances a window by ‘sliding’ it downstream by some no of tuples.
  - Tumble: consecutive windows don’t have overlap
  - Latch: maintain internal state between window.
  - Resample: produce synthetic stream.
- Non-windowed: single tuple at a time
  - Filter: condition
  - Map: apply a function to every tuple
  - GroupBy: partition incoming tuples across multiple streams to groups
  - Join: pairs tuples from input streams
Quality of Service (QoS) must be provided by the application administrator!

The QoS monitor constantly monitors system performance and activates load shedder (ex. Drop tuples) when it is needed, that is, the system performance is degrading by data overload.

Discussion

- The authors state: "Asking the application administrator to specify a multidimensional QoS function seems impractical. Instead, Aurora relies on a simpler tactic, which is much easier for humans to deal with: for each output stream, we expect the application administrator to give Aurora a two-dimensional QoS graph based on the processing delay of output tuples produced." Does this seem easier? Does it make sense to you?

Real Time Scheduling

- Scheduling decision on QoS is not enough!

Maximize overall QoS + reduce overall end to end tuple execution costs!

But how?

Conclusion

- Aurora Stream Query Processing System

- Designed for Scalability
- QoS-Driven Resource Management
- Continuous and Historical Queries
- Stream Storage Management
- Implemented Prototype

www.cs.brown.edu/research/aurora/
Discussion

- Compare Aurora with distributed databases (e.g., Mariposa) and adaptive query execution systems (e.g., Eddies). These systems have to handle arbitrary data arrival rates, and don’t know in advance how much data they will need to process. How does this differ from the continuous query problem? Which techniques are common to both?