Load Shedding
in a Data Stream Manager

Nesime Tatbul, Uğur Çetintemel, Stan Zdonik

Brown University

Mitch Cherniack  Michael Stonebraker

Brandeis University  M.I.T.
The Overload Problem

- Push-based data sources
- High and unpredictable data rates
- Problem:
  Load > Capacity during spikes

"Load Shedding": eliminating excess load by dropping data
Aurora Data Stream Management System

Aurora Query Network

- **Stream Query Algebra (SQuAl)**
- **Quality of Service**

- **Run-time Operation:**
  - Operator Scheduling
  - Storage Management
  - Performance Monitoring
  - Load Shedding
Load Shedding by Inserting Drops

two types of drops:

Random Drop

Semantic Drop

Drop \( k \% \)

Filter \( P(value) \)
Quality of Service

- Loss-tolerance QoS
  - Utility vs. % delivery
  - Values: 1.0, 0.7

- Value-based QoS
  - Utility vs. values
  - Values: 1.0, 0.4

VLDB 2003 Berlin
Assumptions

- Processor is the main scarce resource
- Operators that produce new attribute values are not considered (e.g., aggregate, map)
- Load Shedder operates independently from Scheduler
**Problem Statement**

\( N \): query network  \hspace{1cm} \( C \): processing capacity  
\( I \): set of input streams

when \( \text{Load}(N(I)) > C \), transform \( N \) to \( N' \) such that  
- \( \text{Load}(N'(I)) < C \)  
- \( \text{Utility}(N(I)) - \text{Utility}(N'(I)) \) is minimized

**key questions:**
- when to shed load?
- where to shed load?
- how much load to shed?
- which tuples to drop?
Talk Outline

- Introduction
- Technical Overview
- Experimental Results
- Related Work
- Conclusions and Future Work
Algorithmic Overview

evaluate the Query Network Load

IF Load > Capacity
    insert Drops to the Query Network

ELSE IF Load < Capacity
    AND Drops in the Query Network
    remove Drops from the Query Network

read stats, network
modify network

System Catalog

Queries
Statistics

Drop Insertion Plans

look up
Load Evaluation

“Load Coefficients” for each input stream

\[ L_i = \sum_{j=1}^{n} \left( \prod_{k=1}^{j-1} Sel_k \right) \times Cost_j \] (CPU cycles per tuple)

Total Load at run time

\[ \sum_{i=1}^{m} L_i \times R_i \] (CPU cycles per time unit)
## Load Shedding Road Map (LSRM)

<table>
<thead>
<tr>
<th>Excess Load</th>
<th>Drop Insertion Plan</th>
<th>QoS Cursors</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td><img src="10%25" alt="Diagram" /></td>
<td><img src="10%25" alt="Diagram" /></td>
</tr>
<tr>
<td>20%</td>
<td><img src="20%25" alt="Diagram" /></td>
<td><img src="20%25" alt="Diagram" /></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>300%</td>
<td><img src="300%25" alt="Diagram" /></td>
<td><img src="300%25" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- **Shed Less!**
- **Shed More!**
Constructing the LSRM

1. identify potential “drop locations”
2. sort the drop locations
3. apply the drop locations in order:
   - *insert* one unit of drop
   - if semantic drop, determine the filter predicate
   - create an LSRM entry
Drop Locations

- early drops save more cycles
- drops before splits cause more utility loss
**Best Drop Location**

- **Goal**: maximize cycle gain, minimize utility loss

\[
G(x) = \begin{cases} 
R \times (x \times L - D) & \text{if } x > 0 \\
0 & \text{otherwise}
\end{cases}
\]

- **Loss/Gain Ratio**

\[
\frac{-dU(x)}{dx} = \frac{\text{negative slope of } U(x)}{dG(x)/dx} = \frac{R \times L}{R \times (x \times L - D)}
\]

The smaller, the better!
From Values to % Delivery

- “when, where, how much?” have the same answer for Random and Semantic Load Shedding
- **Trick:** translation between QoS functions
- **Assumption:** drop the lowest utility values first

---

![Graphs showing utility and relative frequency](attachment:graphs.png)
Determining the Filter Predicate

Filter Predicate: 
\[ value \geq 25 \]
Talk Outline

- Introduction
- Technical Overview
- Experimental Results
- Related Work
- Conclusions and Future Work
Experimental Study

Setup:
- streams: uniformly distributed integer values
- networks: a mix of filters and unions
- QoS: value-based QoS, range utilities chosen from a Zipf distribution

Metrics used:
- percent loss in total average utility on
  - loss-tolerance QoS (*Tuple Utility Loss*)
  - value-based QoS (*Value Utility Loss*)
Random-LS beats Admission Control

![Graph showing comparison between Input-Uniform and Random-LS. The x-axis represents mean rate with values ranging from 0.1 to 0.5, and the y-axis represents % tuple utility loss with values ranging from 0 to 70. The graph indicates that Random-LS outperforms Input-Uniform across different loads, with Load ~20% and Load ~300%.]
Semantic-LS beats Admission Control

![Graph showing comparison between Input-Uniform and Semantic-LS]
Semantic-LS beats Random-LS

![Graph showing comparison between Semantic-LS and Random-LS]

- **mean rate:**
  - $r=0.15$
  - $r=0.20$
  - $r=0.25$

- **value utility loss ratio (random-LS/semantic-LS):**

- **skew in utility values (theta):**
  - 0
  - 0.2
  - 0.5
  - 0.8
  - 0.99
Performance for Networks with Sharing

utility skew:
- theta=0.99
- theta=0.5
- theta=0

excess load (%)
tuple utility loss ratio (input-uniform/random-1s)

VLDB 2003 Berlin
Talk Outline

- Introduction
- Technical Overview
- Experimental Results
- Related Work
- Conclusions and Future Work
Relevance to Existing Work

- Congestion control in networks
- Multimedia streaming
- Approximate query answering
- Data stream processing
  - sampling, shedding on aggregates STREAM [MW+03, BDM03]
  - approximate join processing [DGR03]
  - adjusting rates to windowed joins [KNV03]
Talk Outline

- Introduction
- Technical Overview
- Experimental Results
- Related Work
- Conclusions and Future Work
Highlights

- An end-to-end solution for detecting and resolving overload in data stream systems
- Utility loss minimization for networks of queries which share resources and processing
  ➔ semantic utility as well as quantity-based utility
- Static drop insertion plans, dynamic instantiation
Future Directions

- Handling complex operators
  - joins and aggregates
- Other resource limitations
  - memory - windowed operators
  - bandwidth - Aurora*
  - power - at sensor level
More Information

- Aurora Web Page:  
  http://www.cs.brown.edu/research/aurora

- Email:  
  tatbul@cs.brown.edu

- Demo