A Storage-centric Load Management System for Real-time Update Streams
Alexandru Moga and Nesime Tatbul, Systems Group, ETH Zurich, Switzerland

Streaming Applications with Update Semantics
Both data streams and continuous queries can exhibit update semantics...

- **example 1: Financial services**
  - Stream: Trades(time, symbol, price, volume)
  - Quotes(time, symbol, bid, ask, askVol)
  - Query: For each symbol, continuously update the total trade volume and average trading price of the last 10 minutes.

- **example 2: Road traffic monitoring**
  - Stream: Positions(time, objectID, x, y, speed)
  - Query: For each object, continuously update the number of cars in range R of the object in the last minute.

Challenges: Controlled Load Shedding
Update-aware load shedding via subset-based approximation

- **Window Buffers**
  - Keep the most recent window
  - Shed older window tuples using windowing information (size w, slide s)
  - Tuple marking scheme

- **Window updates vs. Window random drops**

- **EAGER vs. LAZY window updates**

Key Load Management (KLM) Update-aware load shedding
- Update semantics: keep the most recent update (in-place updates)
- Query semantics: represent the update unit (e.g., tuple, window of tuples)

Further details
- The UpStream system is a prototype implementation on top of the Borealis SPE.
- Website: http://www.systems.ethz.ch/research/projects/upstream/

Staleness intrinsically to applications with update semantics

Key insight: Exploit update semantics!

- Shed load to keep up with new updates
- Push update semantics upstream

The Problem
Update semantics: the latest result is all that really matters

Staleness: how out-of-date results are compared to latest data

“How to minimize staleness of results for streaming applications with update semantics under conditions of high load?”

Key Scheduling (KS)
- Order keys by earliest enqueue time
  - Group stream tuples by update key
  - Order update keys for processing

Memory Management (MM)
- On-demand paging
  - Early vs. late garbage collection

Staleness
(measured in time per update key, e.g., symbol or object)

Uniform update rates: IN-PLACE Key Scheduling
- Order keys by earliest enqueue time
- Fair Queuing with in-place updates
- Bounded average staleness per key
- Optimal performance for uniform update rates

Non-uniform update rates: LINECUTTING Key Scheduling
- Update frequencies are rarely uniform (e.g., stock price updates)
- IN-PLACE treats all keys “too fairly”
- Idea: Adaptively promote slow updating keys

The UpStream system is a prototype implementation on top of the Borealis SPE.
Website: http://www.systems.ethz.ch/research/projects/upstream/

Staleness is intrinsic to applications with update semantics

Challenges: Controlled Load Shedding

- Update-aware load shedding via subset-based approximation

- Window Buffers
  - Keep the most recent window
  - Shed older window tuples using windowing information (size w, slide s)
  - Tuple marking scheme

- Window updates vs. Window random drops

- EAGER vs. LAZY window updates

Further details
- The UpStream system is a prototype implementation on top of the Borealis SPE.
- Website: http://www.systems.ethz.ch/research/projects/upstream/

Staleness
(measured in time per update key, e.g., symbol or object)

Uniform update rates: IN-PLACE Key Scheduling
- Order keys by earliest enqueue time
- Fair Queuing with in-place updates
- Bounded average staleness per key
- Optimal performance for uniform update rates

Non-uniform update rates: LINECUTTING Key Scheduling
- Update frequencies are rarely uniform (e.g., stock price updates)
- IN-PLACE treats all keys “too fairly”
- Idea: Adaptively promote slow updating keys

Further details
- The UpStream system is a prototype implementation on top of the Borealis SPE.
- Website: http://www.systems.ethz.ch/research/projects/upstream/