

# AURORA: A Data Stream Management System

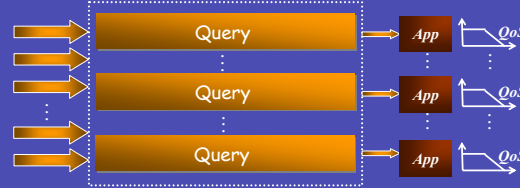
D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, C. Erwin, E. Galvez, M. Hatoun, J. Hwang, A. Maskey, A. Rasin, J. Salz, A. Singer, M. Stonebraker, N. Tatbul, R. Tibbets, Y. Xing, R. Yan, S. Zdonik

A Brandeis, Brown, MIT Production (<http://www.cs.brown.edu/research/aurora>)

## Stream-based Monitoring Applications

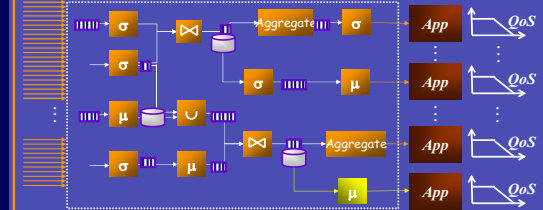
- Monitoring applications:
  - A new class of apps that require timely processing of large volumes of continuous data streams
  - E.g., tracking/monitoring services, financial analysis, sensor networks
  - Traditional DB models are inherently ill suited for these apps
    - Pull- vs. push-based architecture
    - Real-time response requirements
    - Time-series data
    - Approximate answers
- Aurora is a data-stream processing system that is being designed and implemented to support stream-based monitoring applications

## Aurora from 30,000 Feet



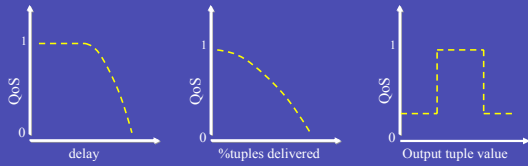
- Applications provide:
  - Queries over input data streams
  - Quality-of-Service (QoS) specifications (specifies the utility of partial or late results)

## Aurora from 100 Feet



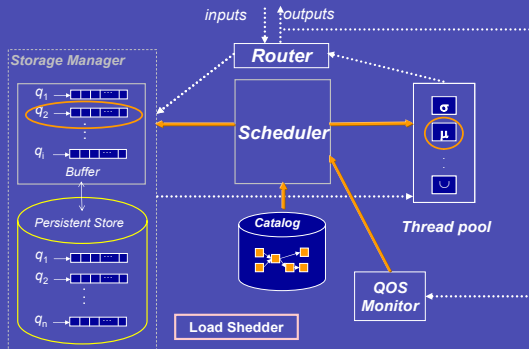
- Boxes  $\rightarrow$  operators
- Arcs  $\rightarrow$  tuple queues
  - can be made persistent via connection points (cylinder icon)
  - queries can access historical data through connection points

## Quality-of-Service

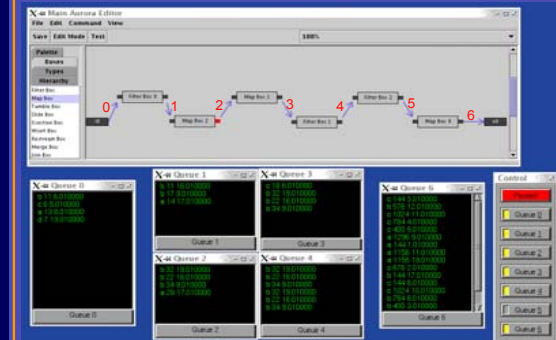


- Per-app QoS specs describe the utility of "imperfect" query results:
  - Delay-based (specify utility of "late" results)
  - Delivery-based (specify utility of "partial" results)
- QoS drives all resource/data management decisions
  - CPU scheduling, storage management, and load shedding, ...

## Run-Time Architecture



## Graphical User Interface



## Operator Scheduling

- Goal: "Minimize per-tuple processing overhead"
  - Diagram showing a sequence of operators A and B. A context switch (red square) occurs between them. The sequence is: A(z), A(y), A(x) followed by B(A(z)), B(A(y)), B(A(x)).
  - Default Operation: ■ = Context Switch
- Overhead reduction via "batching" (aka trains):
  - Tuple Batching: A single operator AB processes the entire batch: B(A(z)), B(A(y)), B(A(x)).
  - Operator Batching: Operator A processes the entire batch first: A(z, y, x), then operator B processes the results: B(A(z), A(y), A(x)).

## Load Shedding

- Drop access load (i.e., tuples) when the system gets overloaded
  - Insert drop operators (skull icon) such that excess load is shed with minimum drop in the perceived QoS
- 
- A diagram showing a query plan with drop operators (skull icons) inserted at various points to shed load. The operators are placed before selection ( $\sigma$ ), join ( $\bowtie$ ), and projection ( $\mu$ ) operators.
- Two types of drop operators:
    - Randomized Drop: Drop random(k %)
    - Semantic Drop: Filter Predicate(value)

## Distributed Processing: Aurora\*

