Window-aware Load Shedding for Aggregation Queries over Data Streams

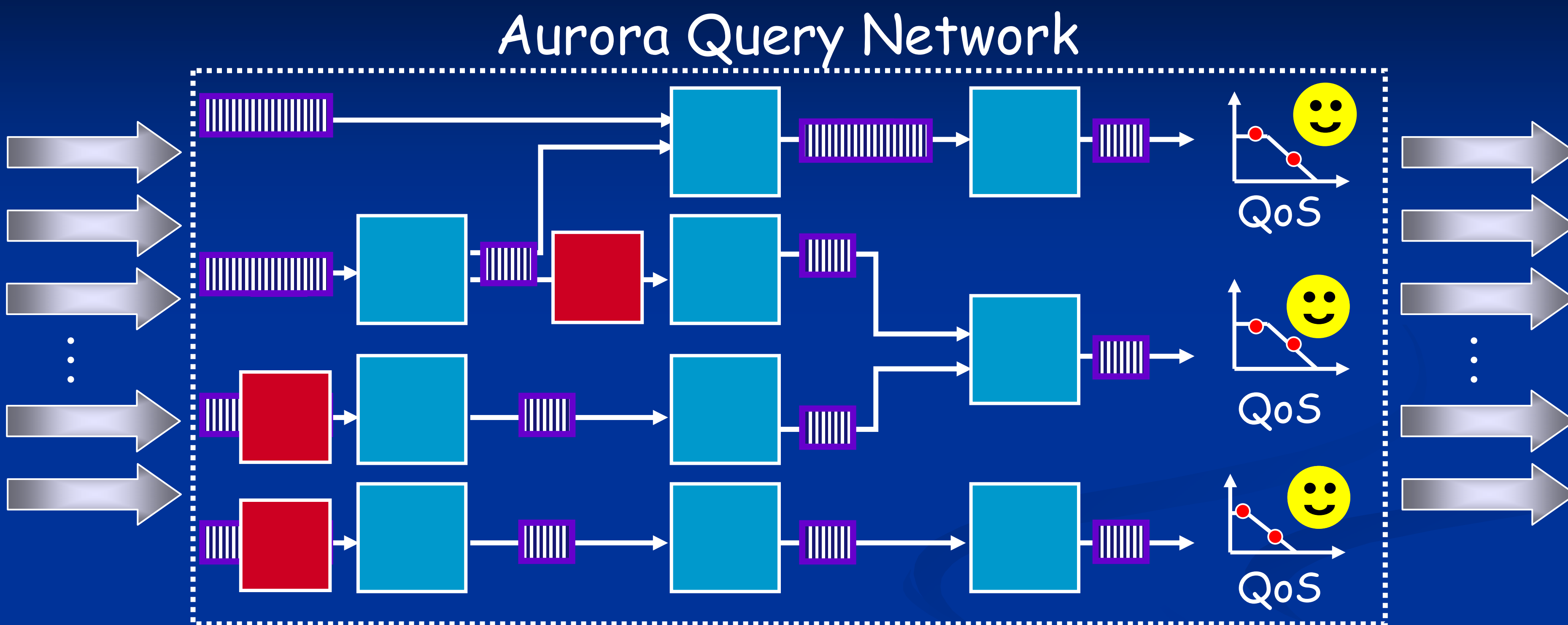
Nesime Tatbul Stan Zdonik

BROWN

Talk Outline

- Background
  - Load shedding in Aurora
  - Windowed aggregation queries
- Window-aware load shedding
- Experimental results
- Related work
- Conclusions and Future directions

Load Shedding in Aurora



- Problem: When load > capacity, latency QoS degrades.
- Solution: Insert drop operators into the query plan.
- Result: Deliver "approximate answers" with low latency.

Subset-based Approximation

- For all queries, the delivered tuple set must be a subset of the original query answer set.
  - Maximum subset measure (e.g., [SIGMOD'03, VLDB'04])
  - Loss-tolerance QoS of Aurora [VLDB'03]

- For each query, the number of consecutive result tuples missed (i.e., gap tolerance) must be below a certain threshold.

Why Subset Results?

- The application may expect to get correct values.
- Missing tuples get updated with more recent ones.
- Preserving subset guarantee anywhere in the query plan helps understand how the error propagates.

- It depends on the application.

Windows

- Finite excerpts of a data stream
- Two parameters: size (ω) and slide (δ)
- Example: **`StockQuote(symbol, time, price)`**

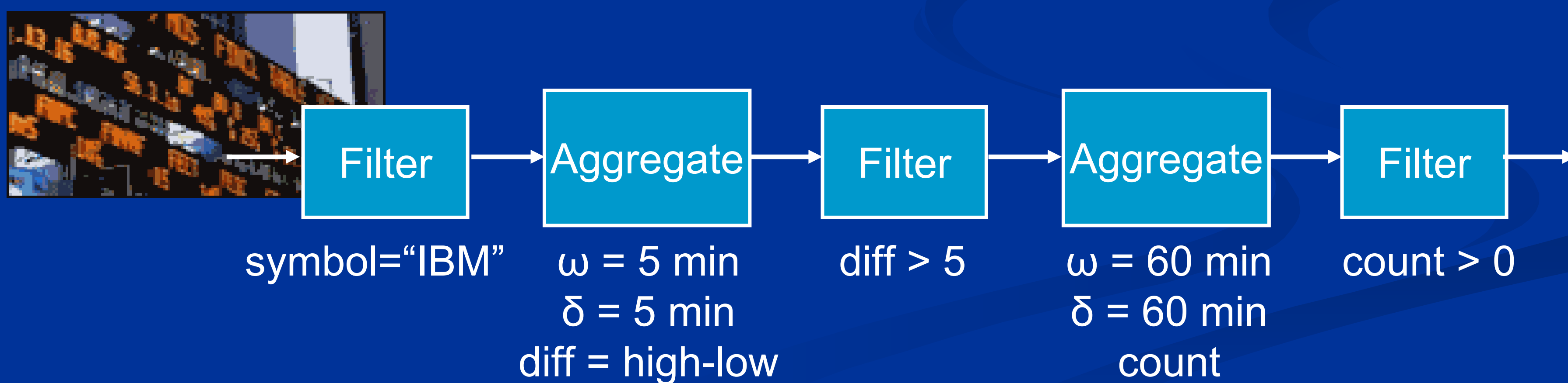
size = 10 min

slide by 5 min

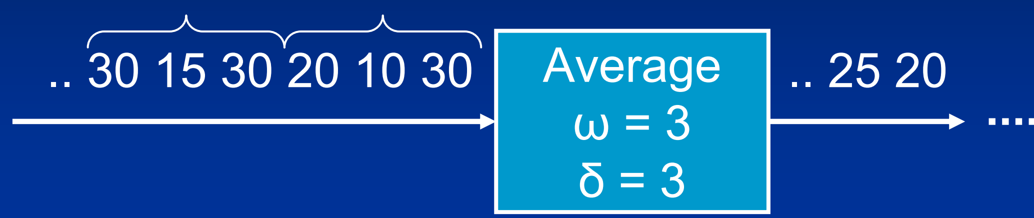
```
("IBM",  10:00, 20)
("INTC", 10:00, 15)
("MSFT", 10:00, 22)
("IBM",  10:05, 18)
("MSFT", 10:05, 21)
("IBM",  10:10, 18)
("MSFT", 10:10, 20)
("IBM",  10:15, 20)
("INTC", 10:15, 20)
("MSFT", 10:15, 20)
```

Windowed Aggregation

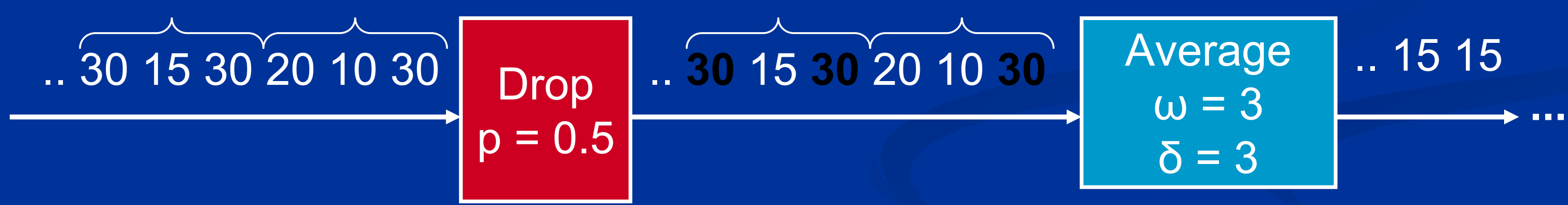
- Apply an aggregate function on the window
  - Average, Sum, Count, Min, Max
  - User-defined function
- Can have grouping, nesting, and sharing
- Example:



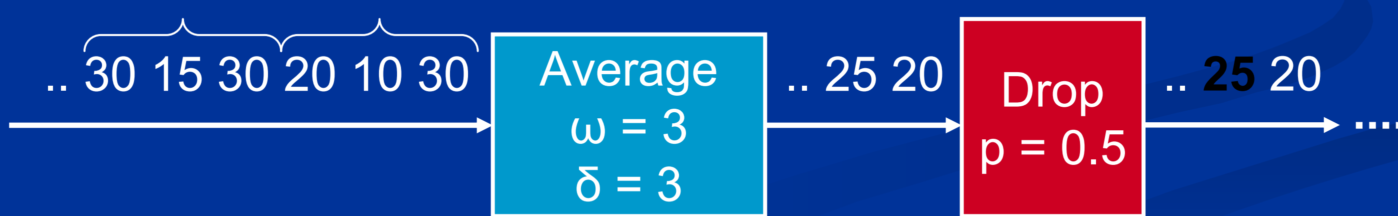
Dropping from an Aggregation Query
Tuple-based Approach



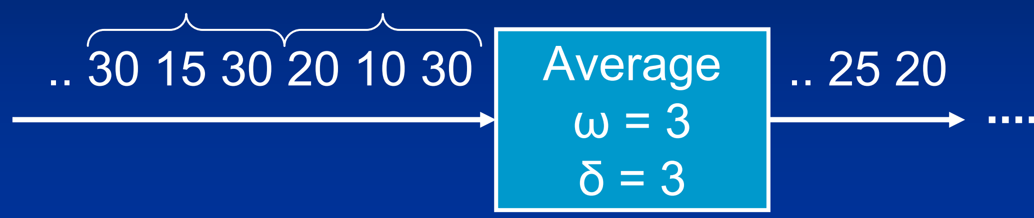
- Drop before : non-subset result of nearly the same size



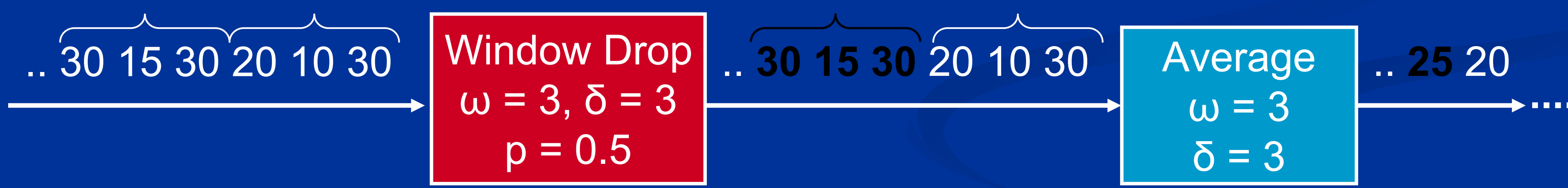
- Drop after : subset result of smaller size



Dropping from an Aggregation Query
Window-based Approach



- Drop before : subset result of smaller size



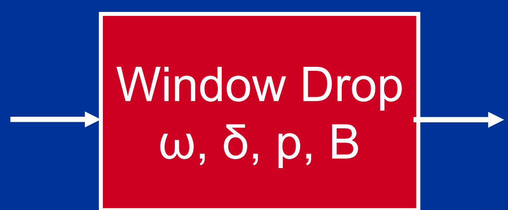
"window-aware load shedding"

Window Drop Operator

- Functionality:
  - Attach window keep/drop specifications into tuples.

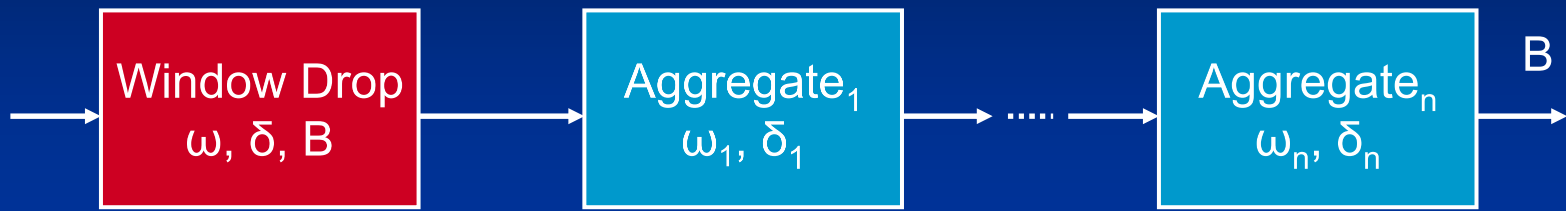
Window Specification	Meaning
-1	Don't care.
0	Drop the window.
T (T > 0)	Keep the window. Keep all tuples with timestamp < T.

  - Discard tuples that can be early-dropped.
- Key parameters:

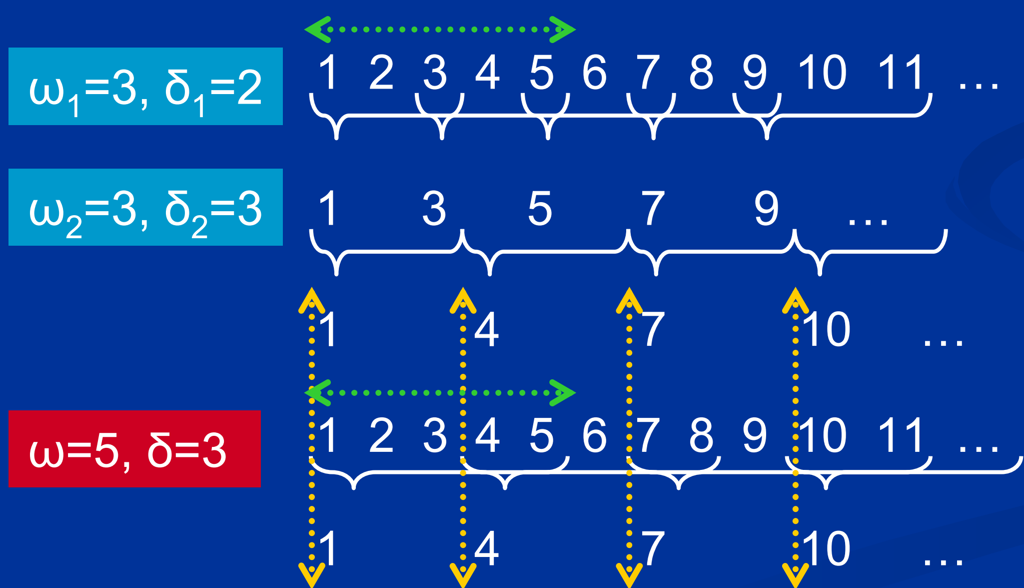


ω: window size
δ: window slide
p: drop probability (one per group)
B: batch size (one per group)

Window Drop for Nested Aggregation (Pipeline Arrangements)



- Example:

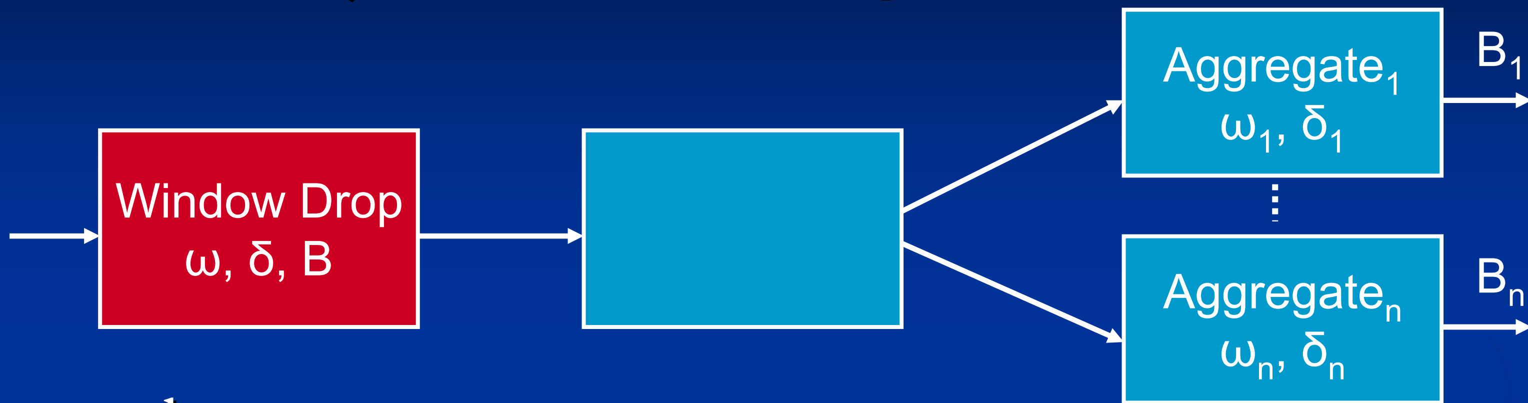


$$\omega = \sum_{i=1}^{n} \omega_i - (n-1)$$

$$\delta = \delta_n$$

$$B = B$$

Window Drop for Shared Aggregation (Fan-out Arrangements)



- Example:

$$\omega = lcm(\delta_1, \ldots, \delta_n) + \max_{i=1}^{n}\{extent(A_i)\}$$

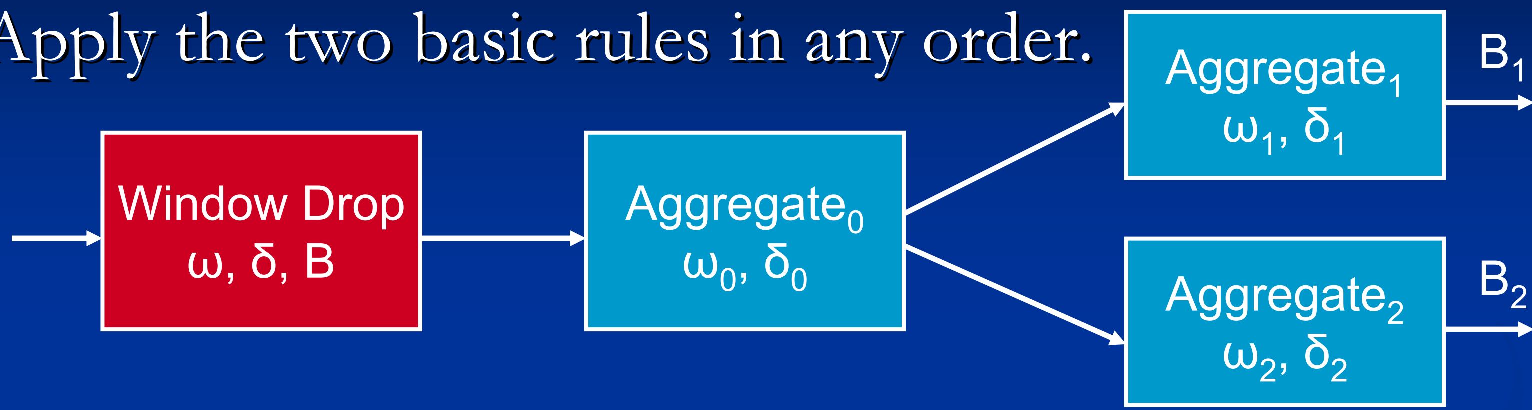
$$\text{where } extent(A_i) = \omega_i - \delta_i$$

$$\delta = lcm(\delta_1, \ldots, \delta_n)$$

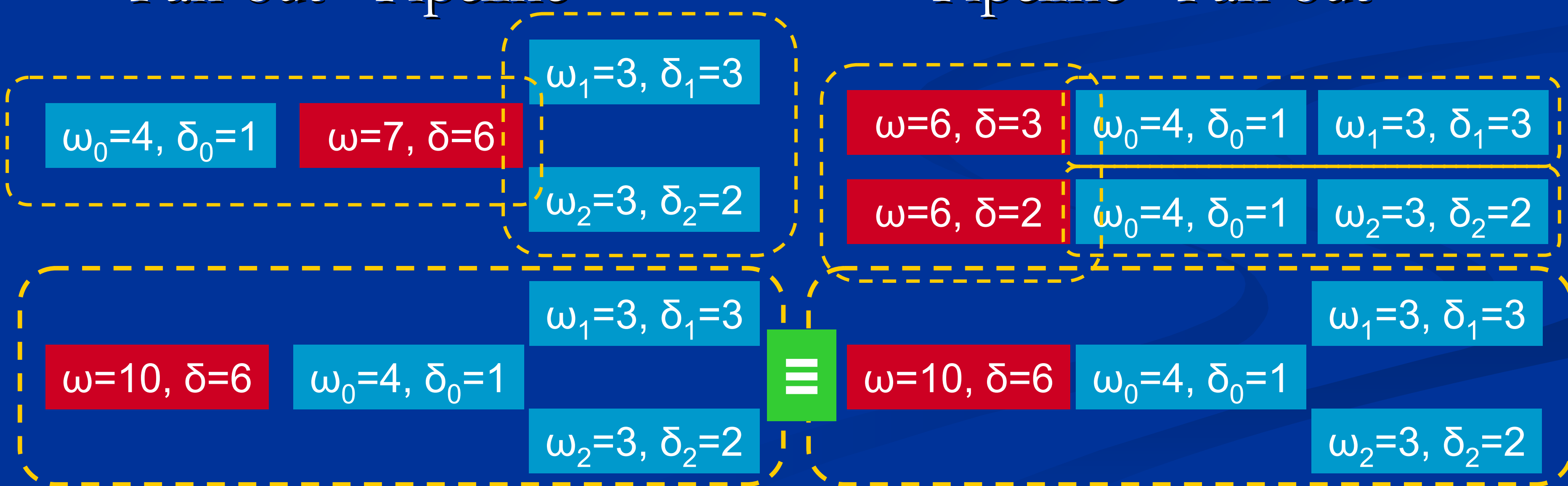
$$B = \min_{i=1}^{n}\left\{\frac{B_i}{lcm(\delta_1, \ldots, \delta_n)/\delta_i}\right\}$$

Window Drop for Nesting + Sharing (Composite Arrangements)

- Apply the two basic rules in any order.

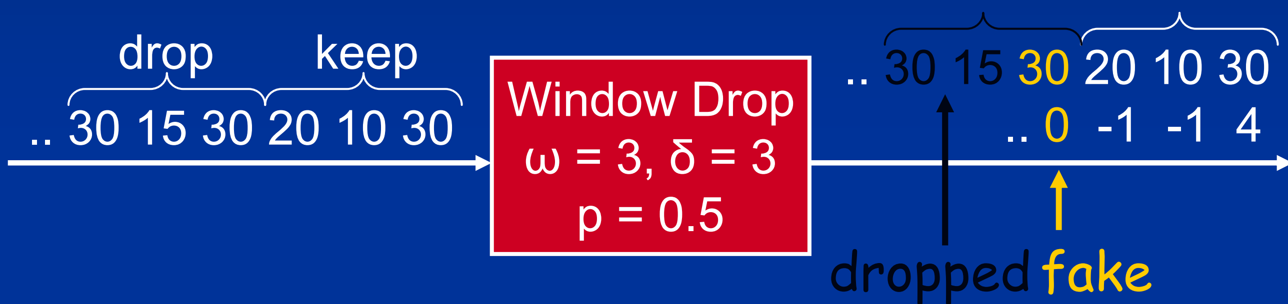


- Fan-out - Pipeline
- Pipeline - Fan-out

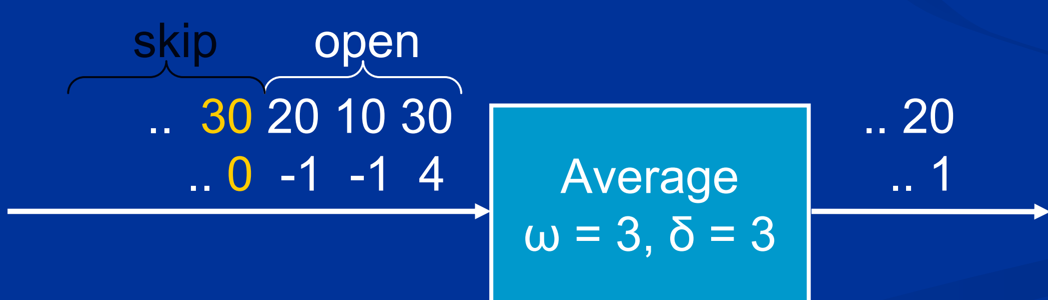


Window Drop in Action

- Mark tuples & apply early drops at the Window Drop.



- Decode the marks at the Aggregate.



Fake Tuples

- Fake tuples carry no data, but only window specs.
- They arise when:
  - the tuple content is not needed as the tuple only indicates a window drop
  - the tuple to mark is
    - originally missing, or
    - filtered out during query processing
- Overhead is low.

Early Drops

- Sliding windows may overlap.
- Window drop can discard a tuple if and only if all of its windows are going to be dropped.
- At steady state, each tuple belongs to ~ ω/δ windows (i.e., for early drops, $B \geq \omega/\delta$).
- If $B < \omega/\delta$, then no need to push window drop further upstream than the first aggregate in the pipeline.

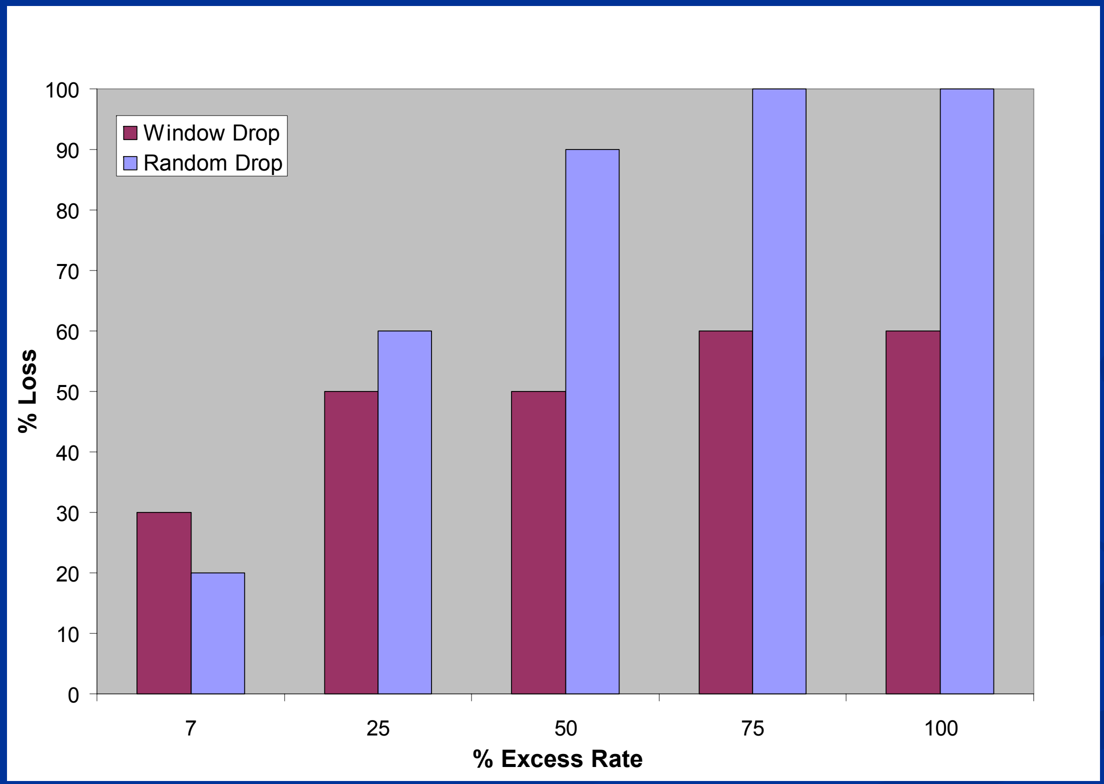
Talk Outline

- Background
  - Load shedding in Aurora
  - Windowed aggregation queries
- Window-aware load shedding
- Experimental results
- Related work
- Conclusions and Future directions

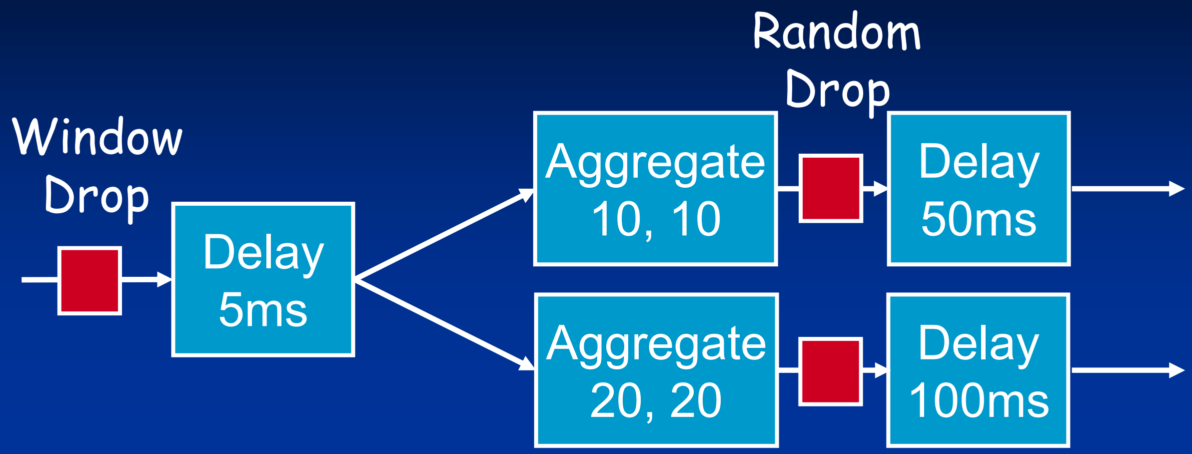
Experiments

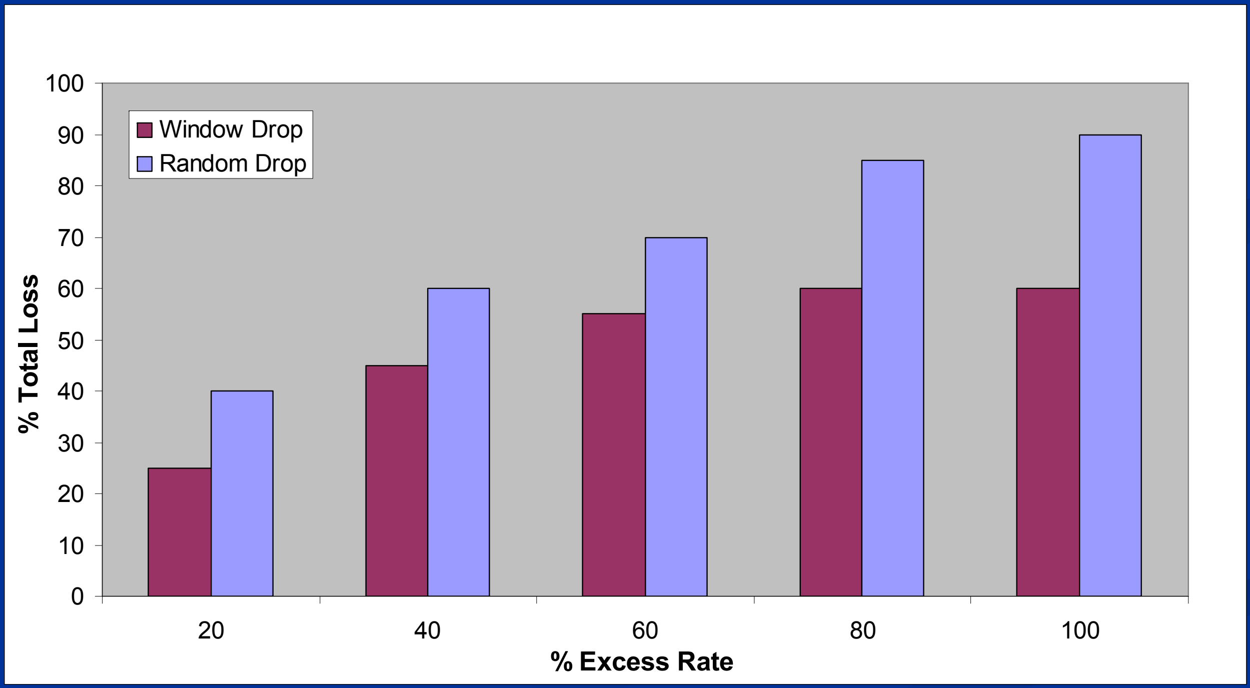
- Setup
  - Single-node Borealis
  - Streams: (time, value) pairs
  - Aggregation queries with "delays"
  - Basic measure: % total loss
- Goals
  - Performance on nested and shared query plans
  - Effect of window parameters
  - Processing overhead

Performance for Nested Plans

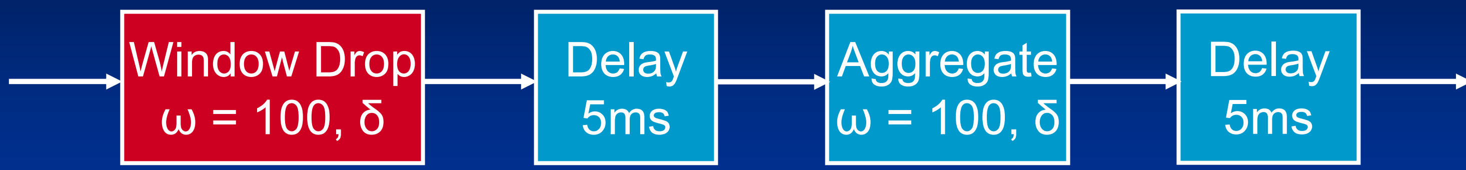


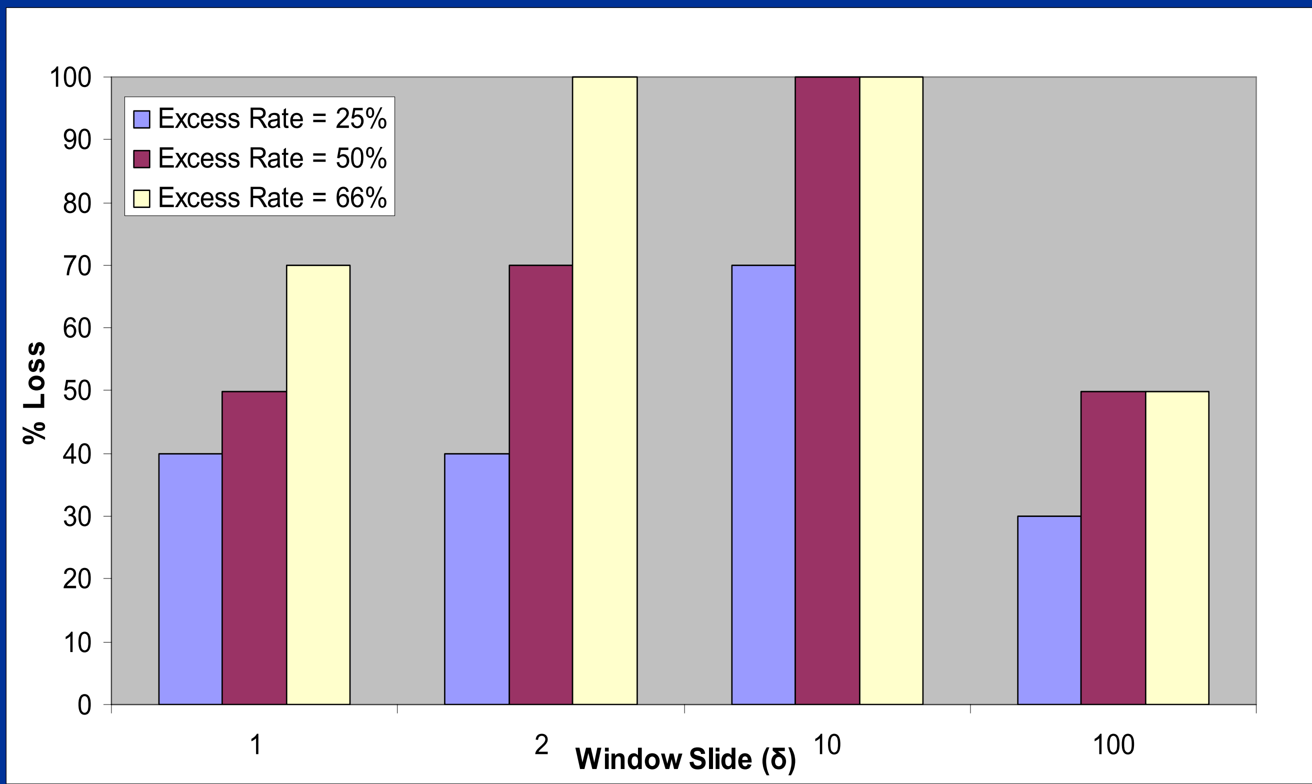
Performance for Shared Plans



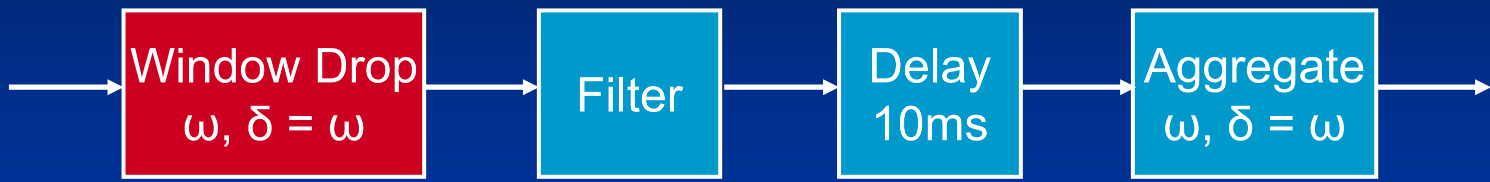


Effect of Window Slide





Processing Overhead



Throughput Ratio (Window-Drop (p=0) / No Window-Drop)

Window size (ω)	Selectivity = 1.0	Selectivity = 0.5
25	0.99	0.96
50	0.99	0.98
75	1.0	0.98
100	1.0	1.0

Related Work

- Load shedding for aggregation queries
  - Statistical approach of Babcock et al [ICDE'04]
- Punctuation-based stream processing
  - Tucker et al [TKDE'03], Li et al [SIGMOD'05]
- Other load shedding work (examples)
  - General: Aurora [VLDB'03], Data Triage [ICDE'05]
  - On joins: Das et al [SIGMOD'03], STREAM [VLDB'04]
  - With optimization: NiagaraCQ [ICDE'03, SIGMOD'04]
- Approximate query processing on aggregates
  - Synopsis-based approaches, online aggregation [SIGMOD'97]

Conclusions

- We provide a **Window-aware Load Shedding** technique for a general class of aggregation queries over data streams with:
  - sliding windows
  - user-defined aggregates
  - nesting, sharing, grouping
- **Window Drop** preserves window integrity, enabling:
  - easy control of error propagation
  - subset guarantee for query results
  - early load reduction that minimizes error

Future Directions

- Prediction-based load shedding
  - Can we guess the missing values?
  - Statistical approaches vs. Subset-based approaches
- Window-awareness on joins
- Memory-constrained environments
- Distributed environments

Questions?

Decoding Window Specifications

Window Start	Window Spec	Keep Boundary	To Do
✓	T	*	Open window. Set boundary = T – (ω – 1) Set spec = T – (ω – 1)
✓	0 -1	Within	Open window. Set boundary = t + ω (if >)
✓	0 -1	Beyond	Skip window.
✗	T	*	Set boundary = T – (ω – 1) Set spec = T – (ω – 1) Mark as "fake tuple".
✗	0	*	Mark as "fake tuple".
✗	-1	*	Ignore.