

Freshness-Aware Scheduling of Continuous Queries in the Dynamic Web

Mohamed A. Sharaf
Alexandros Labrinidis
Panos K. Chrysanthis
Kirk Pruhs

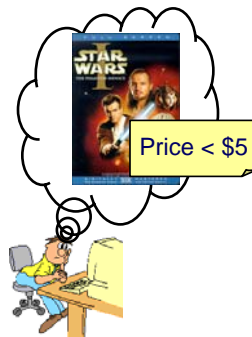
Advanced Data Management Technologies Lab
Department of Computer Science
University of Pittsburgh



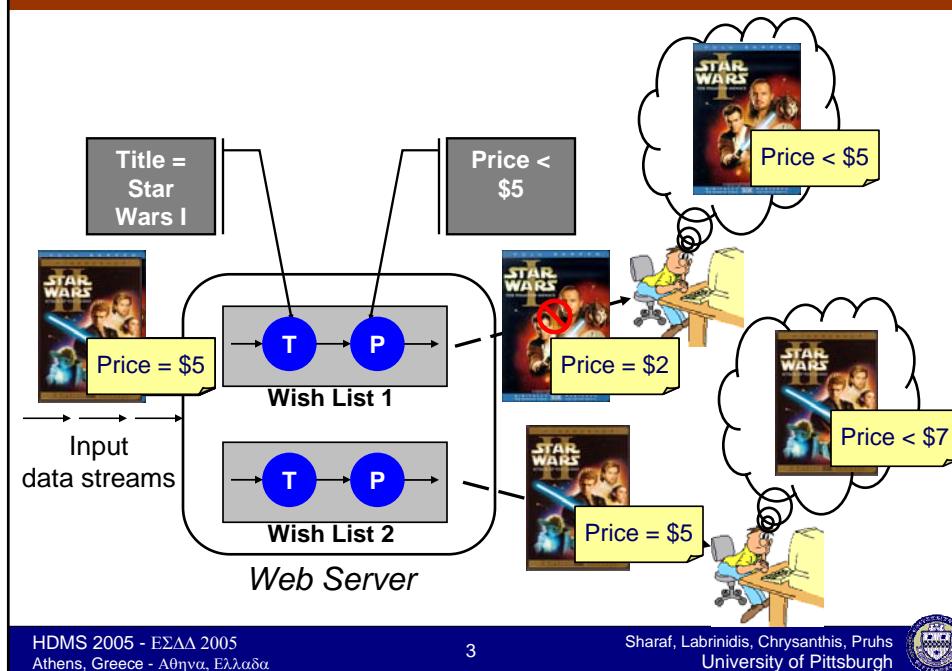
HDMS 2005



Motivation



Motivation



Continuous Queries

- A continuous query is a long-running standing query whose execution is triggered with the arrival of new data
- The output of a continuous query is a continuous data stream (e.g., e-mails or update personalized Web page)
- A Web server should propagate updates to users as soon as they become available, however delays occur due to:
 - 1) Time spent by a query **processing updates**
 - 2) Time spent by a query **waiting** to be executed

Scheduling Multiple Continuous Queries

- The execution order of continuous queries determines the overall **behavior** of the system. For example:
 - In Aurora: the Minimum Latency scheduler reduces **response time** [Carney et. al., VLDB'03]
 - In STREAM: the Chain scheduler minimizes **memory usage** [Babcock et. al., SIGMOD'03]
- Problem Statement:
 - *Devise a policy for scheduling the execution of multiple continuous queries (MCQ) with the objective of maximizing the overall **quality of data (QoD)** of output data streams*



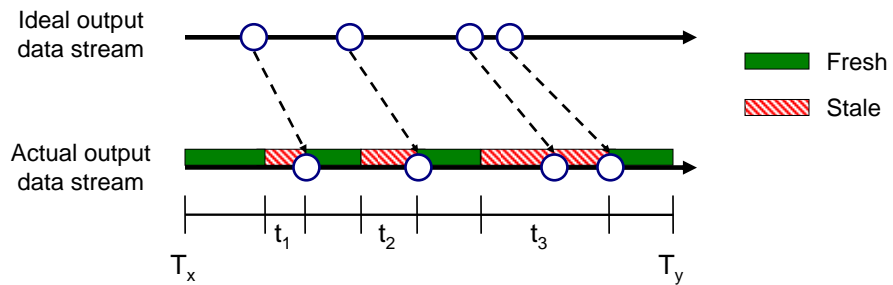
Outline

- Motivation
- **Quality of Data (QoD)**
- Freshness-Aware Scheduling of MCQ (FAS-MCQ)
- Experimental Evaluation
- Conclusions and Future Work



Quality of Data

- QoD based on freshness (deviation from the ideal)
- At any time instance, the output data stream is fresh when it matches the ideal one, otherwise it is stale



- $Delay (t_i) = \text{wait time} + \text{processing time}$
- $Freshness = 1 - (t_1 + t_2 + t_3) / (T_y - T_x)$

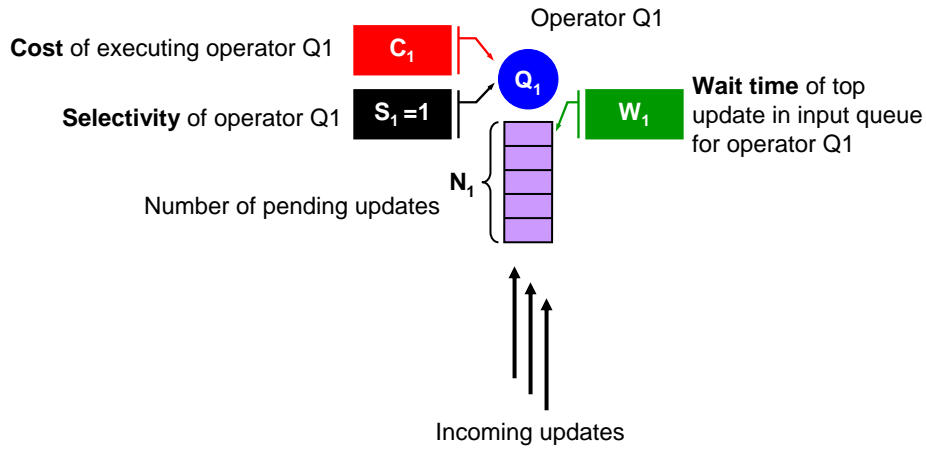


Outline

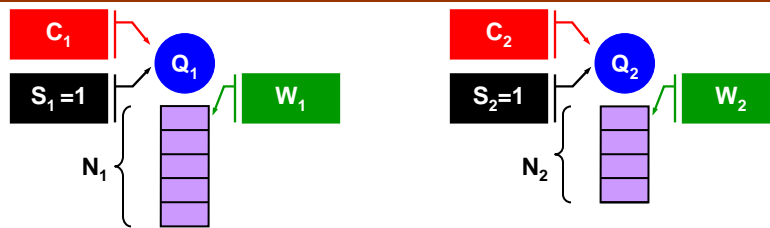
- Motivation
- Quality of Data (QoD)
- **Freshness-Aware Scheduling of MCQ (FAS-MCQ)**
- Experimental Evaluation
- Conclusions and Future Work



How updates are processed



Freshness-Aware Scheduling of MCQ (FAS-MCQ)



- Compute loss in freshness (L) under two policies:

- Policy X: first Q_1 then Q_2
- Policy Y: first Q_2 then Q_1

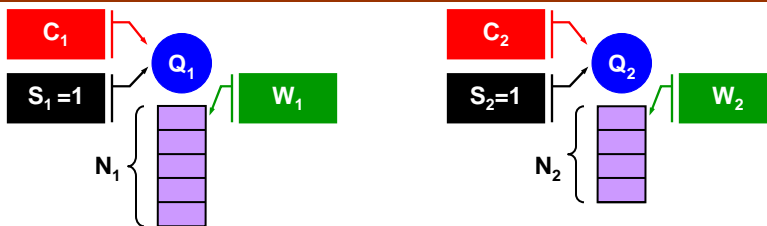
- $$L_X = \overbrace{(W_1 + N_1 * C_1)}^{Q_1\text{'s loss}} + \overbrace{(W_2 + N_2 * C_2 + N_1 * C_1)}^{Q_2\text{'s loss}}$$

Current loss
in freshness

Loss until
processing
pending tuples

Loss due to
waiting for Q_1 to
finish execution

Freshness-Aware Scheduling of MCQs



- Under policy X: first Q_1 then Q_2
 - $L_X = (W_1 + N_1 * C_1) + (W_2 + N_1 * C_1 + N_2 * C_2)$
- Under policy Y: first Q_2 then Q_1
 - $L_Y = (W_2 + N_2 * C_2) + (W_1 + N_2 * C_2 + N_1 * C_1)$
- For $L_X < L_Y \Rightarrow N_1 * C_1 < N_2 * C_2$

$$\text{Priority of } Q_i = 1/(N_i * C_i)$$



Impact of Selectivity

- A query is a tree of operators
- Each query operator is associated with:
 - Cost (c): processing time
 - Selectivity (s): probability of producing an output after processing an input update



- **Maximum cost:**

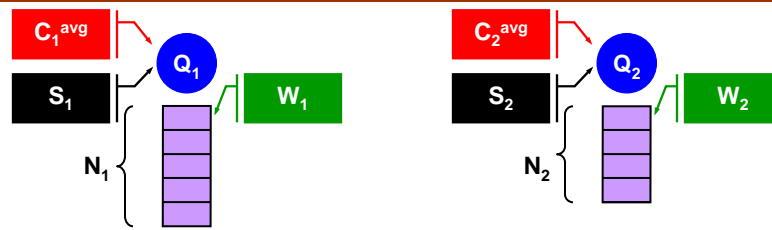
$$C = C_1 + C_2 + C_3$$
- **Total selectivity:**

$$S = S_1 * S_2 * S_3$$
- **Average/Expected Cost:**

$$C^{avg} = C_1 + (C_2 * S_1) + (C_3 * S_1 * S_2)$$



Selectivity-Aware FAS-MCQ



- Priority of $Q_i = 1/(N_i * C_i^{avg})$
- But we need to consider *selectivity*:
 - If $S_i = 0$, no appending and the output data stream is fresh
 - If $S_i = 1$, appending and the output data stream is stale
- Compute the staleness probability ($P_i = 1 - (1-S_i)^{N_i}$)

$$\text{Priority of } Q_i = P_i / (N_i * C_i^{avg})$$



Summary

$$\text{Priority of } Q_i = P_i / (N_i * C_i^{avg})$$

- FAS-MCQ behaves as follows:
 - If all queries have the same P (*staleness probability*) and N (*number of pending updates*), then FAS-MCQ selects the query with the **lowest cost**
 - If all queries have the same P and C (*expected cost*) then FAS-MCQ selects the query with the **lowest number of pending updates**
 - If all queries have the same N and C , then FAS-MCQ selects the query with the **highest staleness probability**



Intuitions underlying FAS-MCQ

$$\text{Priority of } Q_i = P_i / (N_i * C_i^{avg})$$

- The priority of a query increases if it has:
 - Small processing cost (C_i),
 - Small number of pending updates (N_i),
 - High staleness probability (P_i)



Outline

- Motivation
- Quality of Data (QoD)
- Freshness-Aware Scheduling of MCQ (FAS-MCQ)
- **Experimental Evaluation**
- Conclusions and Future Work



Simulation Testbed

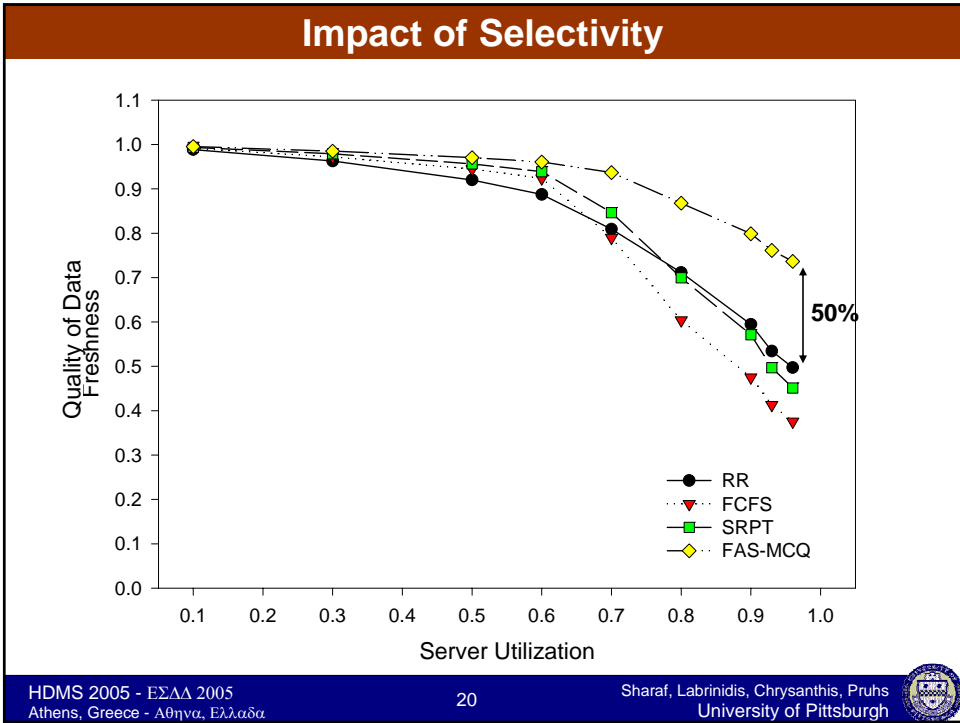
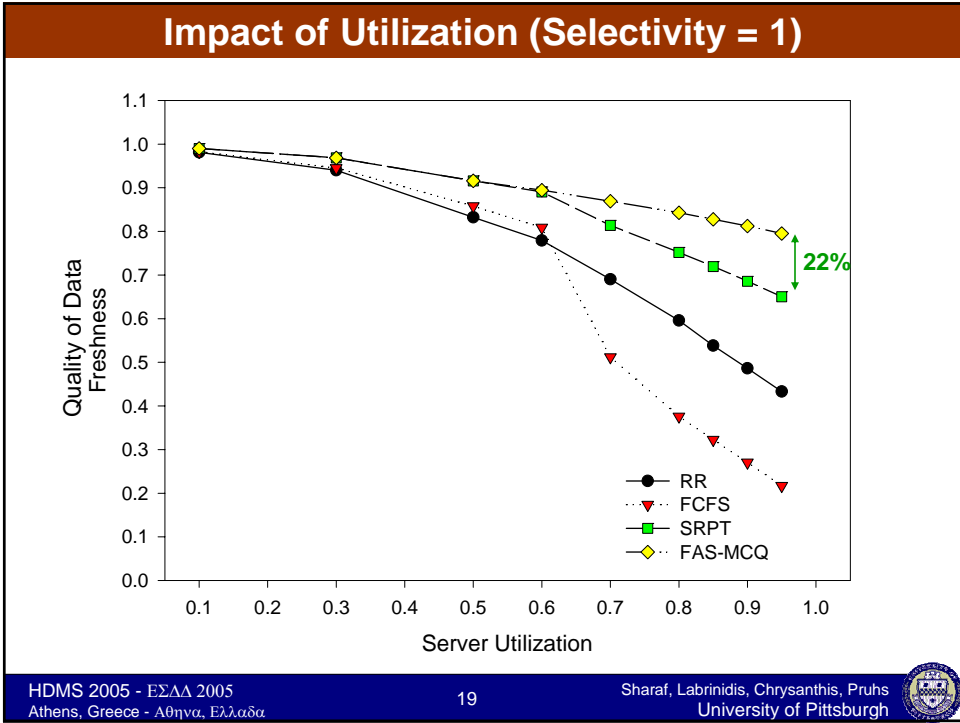
- Simulated the execution of 250 continuous queries with variable costs and selectivities
- 10 input data streams following Poisson distribution
- Half of the input streams are bursty
- Experiments to show:
 - Impact of utilization,
 - Impact of Selectivity,
 - (Impact of burstiness),
 - Fairness



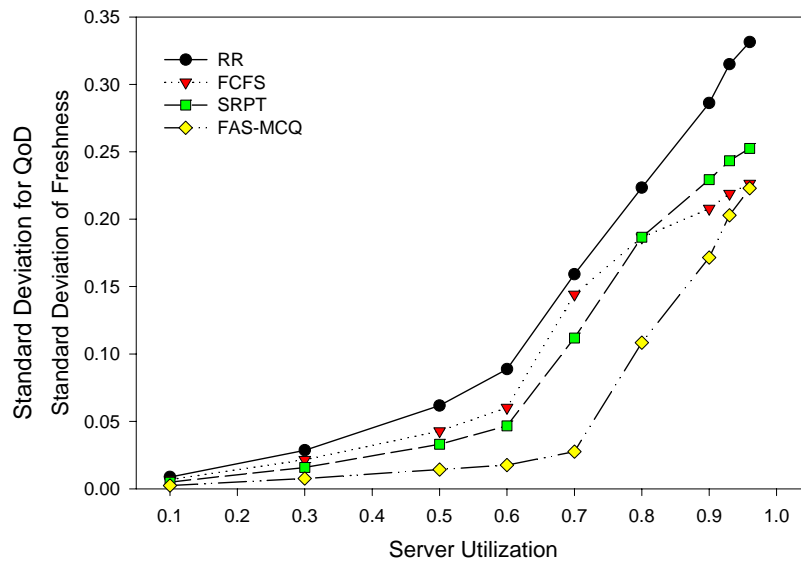
Scheduling Algorithms

- **FAS-MCQ:**
Freshness-Aware Scheduling of Multiple Continuous Queries
- **RR:**
Round-Robin (Aurora)
- **SRPT:**
Shortest Remaining Processing Time
- **FCFS:**
First-Come First-Served





Fairness



Conclusions

- We proposed a policy for **freshness-aware scheduling** of multiple continuous queries.
- Our policy exploits the properties of continuous queries (i.e., **cost and selectivity**) as well as the properties of input data streams (**variability of updates**)
- We showed experimentally that our proposed policy outperforms the traditional scheduling policies
- **Future:** study multi-stream queries and different definitions for QoD



Ευχαριστώ

Ερωτήσεις;

Advanced Data Management Technologies Lab
<http://db.cs.pitt.edu>

