Static Optimization of Conjunctive Queries with Sliding Windows Over Infinite Streams
Ahmed M. Ayad and Jeffrey F. Naughton

Presenter: Maryam Karimzaehgan
mkarimzaehgan@cs.uwaterloo.ca

Outline
- Introduction
- Definition -- Selection and Join Semantics
- The Cost Model
- Load Shedding
- Experiments
- Conclusion
- Discussion Points

Introduction
- The aim of the paper:
  - Find a execution plan that minimizes resource usage when resources are sufficient.
  - Find an execution plan that sheds tuples when resources are insufficient.
- Execution plan is like Queuing Network System
  - Arriving tuples are clients
  - Query operators are servers
- Execution plan is feasible if the system is stable.
- At least one feasible plan exists for feasible query.
**Example – Feasible and Infeasible**

Selectivity

\[ C = 10^3 \]
\[ \alpha = 0.5 \]
\[ \gamma = 0.25 \]
\[ j = 250 \]
\[ j = 500 \]

75% resource utilization for plan A

\[ (\alpha) = 0.5 \]
\[ (\gamma) = 0.25 \]

Plan A - Feasible

Arrival Rate

\[ C = 2 \times 10^4 \]
\[ \alpha = 0.25 \]
\[ \gamma = 0.05 \]
\[ j = 1000 \]
\[ j = 1000 \]

Plan B - Infeasible

Output rate is its input rate multiplied by its selectivity

---

**Introduction**

- If plan is infeasible → load shedding → Random dropping of tuples
- The goal of load shedding is the plan that maximizes the output rate
- Two different problems:
  - Optimal placement of drop boxes and optimal setting of sampling rate
  - Choice of plan to shed load from

  *If query is feasible* → find feasible plan with lowest resource utilization
  *If query is infeasible* → search the plan that yields maximum output rate when tuples are dropped from it

---

**Simplifying Assumptions**

- Timestamp is unique for any data stream and timestamps are assigned by the system for each tuple upon arrival
- No out of order arrival
- No relational tables involved in the query
- Static optimization → Rates of input streams are slow changing (steady state condition)
- Enough memory to hold the buffering requirements for any query plan.
Outline
- Introduction
- Definition -- Selection and Join Semantics
- The Cost Model
- Load Shedding
- Experiments
- Conclusion
- Discussion Points

Selection and Join Semantics
- A selection (Filter) operator takes a stream as an input and outputs the Stream elements that are the subset of input stream satisfying the selection predicate
- Sliding Window Join Symmetric operator that takes two input streams. For every arriving tuple the operator joins it with the current window contents
The Cost Model

- Selections and projections
  - The number of tuples in a unit time is \( \lambda_i \)
  - The output rate is \( \lambda_i = \lambda_i \cdot \lambda_i \)

- Active Window Size is \( W_i = f \cdot W_i \)

- Joins and Cartesian Products
  - The output rate is \( \lambda_i = \lambda_i + W_i \cdot \lambda_i \)
  - Active Window Size is \( W_i = f \cdot W_i \cdot W_i \)

Processing Constraints

- Stream A, B, C, 10 rows in each stream
- Arrival of Stream A = 10, B = 70, C = 20 tuples/second
- Selectivity of A \& B = 0.5 and A \& C = 0.2

All three plans have the same final output rate
1. Join Operator 0.5 milliseconds → 2000 tuples/second, Plan A: 25% B: 14% C: 20% B is the best choice
2. Join Operator 3 milliseconds → 334 tuples/second, Plan A: 150% B: 84% C: 120% Only B is feasible
3. Join Operator 5 milliseconds → 200 tuples/second, All Plans become infeasible

All three plans have the same final output rate

Outline
- Introduction
- Definition -- Selection and Join Semantics
- The Cost Model
- Load Shedding
- Experiments
- Conclusion
- Discussion Points
Load Shedding

- Random dropping of tuples
- Goal is to maximize the output rate of the approximated query
- Two questions:
  - Given a plan, where the drop box should be placed?
  - Which plan should be chosen for load shedding?
- Choose the best plan when resources were sufficient

Selection Only Queries

- \( n+1 \) possible places to put drop boxes
- To approximate a plan for filtering drop tuples from streaming source before they are processed
- The approximation on a plan with the least cost in order to maximize the output

Join Queries

- A drop box can be put before each of the two inputs
  
  \[ \lambda_x \]
  
  \[ \sum_{i=1}^{n} x_i \]
  
  \[ \lambda_{x_i} \]
  
  \[ \lambda_y \]
  
  \[ \sum_{i=1}^{n} y_i \]
  
  \[ \lambda_{y_i} \]
  
  • Tuples should be dropped from the input streams before being processed by any join operators
Choice of Plan for Load Shedding

- Plan with the lowest resource utilization does not carry over the case of join queries

The plan with the lowest utilization is not always the best for load shedding

- Plan with the lowest resource utilization does carry over the case of join queries

Optimization Framework

- Load shedding should be integrated in the process of optimization

  Optimization Problem
  - Two functions
    - Throughput of the plan
    - Utilization cost of the plan
  - For feasible queries → the goal of optimization is
    - Minimize utilization cost while throughput is at its maximum value
  - For infeasible queries →
    - Maximize throughput while the utilization cost is fixed at its maximum value
The Objective of the Optimization

- To maximize “\( R(P) = \text{throughput/utilization cost} \)"

- Simplest optimization algorithm
  1. generate the set of plans for the query
  2. For each of these plans, compute utilization cost
  3. if utilization cost > 1, insert drop box
  4. compute \( R \)
  5. return the plan that maximizes \( R(P) \)

Heuristic Algorithm

- The algorithm builds a plan bottom up by storing the best plans for successively larger subsets of the input streams.
- Compute the best plan for any subset
- Tests whether subplan is feasible
- If the plan is infeasible
  - tunes the values of drop boxes placed at its input streams
  - stores the subplan with the settings of its drop boxes
- If at any stage, the algorithm places a drop box in front of the stream which had another one from the previous round → combine into one drop box.

Outline

- Introduction
- Definition -- Selection and Join Semantics
- The Cost Model
- Load Shedding
- Experiments
- Conclusion
- Discussion Points
Experiments

- 1000 random queries
- Each query → join of five input stream sources A, B, C, D, E
- Rate of input streams from 10 to 1000 tuples/sec
- Window size and join selectivities are fixed for all queries

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_{A=B}</td>
<td>0.2</td>
<td>100</td>
</tr>
<tr>
<td>f_{A=C}</td>
<td>0.5</td>
<td>300</td>
</tr>
<tr>
<td>f_{B=D}</td>
<td>0.1</td>
<td>500</td>
</tr>
<tr>
<td>f_{B=E}</td>
<td>0.001</td>
<td>1000</td>
</tr>
</tbody>
</table>

The Need for Reoptimization

Experiments – Average Gain

Average Gain is the ratio between the difference of the two throughputs and the lowest utilization throughput.
Experiments – Max Gain

Not to consider queries that did not need reoptimization → max gain

Experiments – Heuristic Optimizer

- At every examined level of resource → optimize every query using optimizer
- For every query → difference of the objective function between the plan found by optimizer and search

Conclusion

- A framework for static optimization of sliding window conjunctive over infinite streams
- Cost model for estimating average resource utilization and output rate of the plan
- Optimization algorithm → resource constraints into the optimization process
- Need for reoptimization
Discussion Points

- Selection only queries and join only queries, what if they have the combination of both?
- If they have multiple queries → resource sharing is an issue
- Average steady rate of arrival of data streams, what if the pattern of changes is not predictable?
- Enough memory to hold the buffering requirements
- Random dropping of tuples, how to do it semantically?
- Experiments in term of accuracy
- Optimizer is not that much accurate at very low resource
- High response time for executing plan for each query

QUESTIONS?