Static Optimization of Conjunctive Queries with Sliding Windows Over Infinite Streams

Ahmed M. Ayad and Jeffrey F.Naughton

Presenter: Maryam Karimzadehgan
mkarimzadehgan@cs.uwaterloo.ca

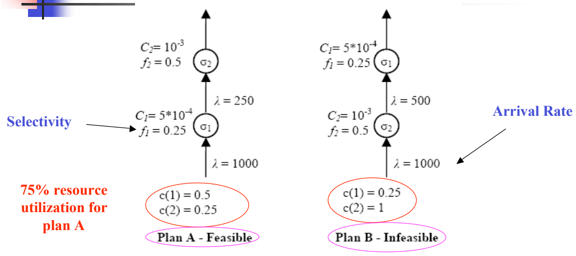
Outline

- 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



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 → 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 if 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

7

Selection and Join Semantics

- **A selection (Filter) operator takes a stream as an input and outputs the Stream → elements 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**

8

Outline

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

9

The Cost Model

- Selections and projections
  - The number of tuples in a unit time is λ_i
    - **The output rate is** $\lambda_o = f.\lambda_i$
    - **Active Window Size is** $W_0 = f.W_i$
- Joins and Cartesian Products
    - **The output rate is** $\lambda_o = f(W_R.\lambda_L + W_L.\lambda_R)$
    - **Active Window Size is** $W_o = f.W_L.W_R$

10

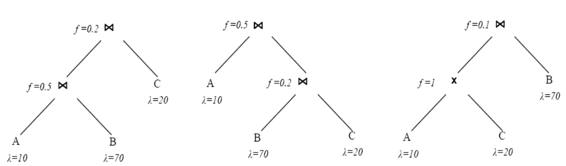
Processing Constraints

```
SELECT A.a, B.b, C.c
FROM      A [ROWS 10]
          B [ROWS 10]
          C [ROWS 10]
WHERE     A.a = B.a
AND       B.b = C.b
```

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

11

Processing Constraints



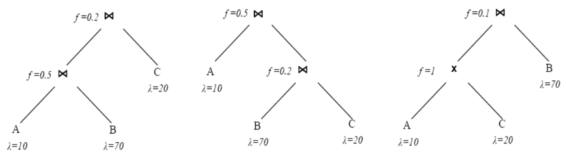
Plan (a)	Plan (b)	Plan (c)
$\lambda_{A\bowtie B} = 0.5*(10*10+70*10) = 400$	$\lambda_{B\bowtie C} = 0.2*(20*10+70*10) = 180$	$\lambda_{A\bowtie C} = 1*(10*10+20*10) = 300$
$W_{A\bowtie B} = 0.5*10*10 = 50$	$W_{B\bowtie C} = 0.2*10*10 = 20$	$W_{A\bowtie C} = 1*10*10 = 100$
$\lambda_{(A\bowtie B)\bowtie C} = 0.2*(400*10+20*50) = 1000$	$\lambda_{A\bowtie (B\bowtie C)} = 0.5*(180*10+20*10) = 1000$	$\lambda_{(A\bowtie C)\bowtie B} = 0.1*(300*10+70*100) = 1000$

All three plans have the same final output rate

12

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



Plan (a) | Plan (b) | Plan (c)

$\lambda_{A\bowtie B} = 0.5*(10*10+70*10) = 400$

$W_{A\bowtie B} = 0.5*10*10 = 50$

$\lambda_{(A\bowtie B)\bowtie C} = 0.2*(400*10+20*50) = 1000$

$\lambda_{B\bowtie C} = 0.2*(20*10+70*10) = 180$

$W_{B\bowtie C} = 0.2*10*10 = 20$

$\lambda_{A\bowtie (B\bowtie C)} = 0.5*(180*10+20*10) = 1000$

$\lambda_{A\bowtie C} = 1*(10*10+20*10) = 300$

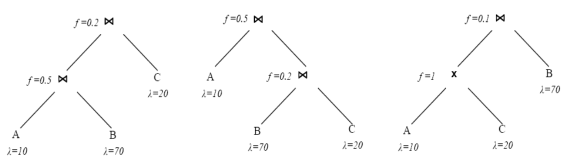
$W_{A\bowtie C} = 1*10*10 = 100$

$\lambda_{(A\bowtie C)\bowtie B} = 0.1*(300*10+70*100) = 1000$

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



Plan (a) | Plan (b) | Plan (c)

$\lambda_{A\bowtie B} = 0.5*(10*10+70*10) = 400$

$W_{A\bowtie B} = 0.5*10*10 = 50$

$\lambda_{(A\bowtie B)\bowtie C} = 0.2*(400*10+20*50) = 1000$

$\lambda_{B\bowtie C} = 0.2*(20*10+70*10) = 180$

$W_{B\bowtie C} = 0.2*10*10 = 20$

$\lambda_{A\bowtie (B\bowtie C)} = 0.5*(180*10+20*10) = 1000$

$\lambda_{A\bowtie C} = 1*(10*10+20*10) = 300$

$W_{A\bowtie C} = 1*10*10 = 100$

$\lambda_{(A\bowtie C)\bowtie B} = 0.1*(300*10+70*100) = 1000$

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

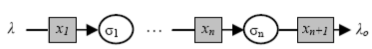
Load Shedding

- **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**

16

Selection Only Queries

- **n+1 possible places to put drop boxes**

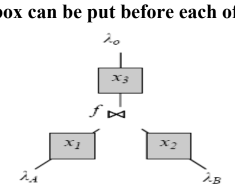


- **To approximate a plan for filtering → drop tuples from streaming source before they are processed**
- **The approximation → on a plan with the least cost in order to maximize the output**

17

Join Queries

- **A drop box can be put before each of the two inputs**

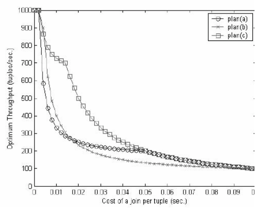


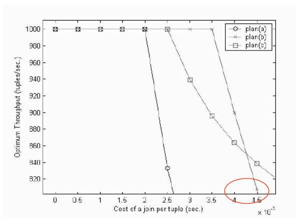
- **Tuples should be dropped from the input streams before being processed by any join operators**

18

Choice of Plan for Load Shedding

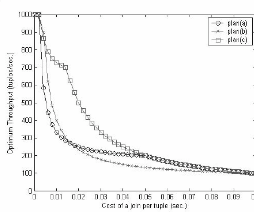
- **Plan with the lowest resource utilization does not carry over the case of join queries**

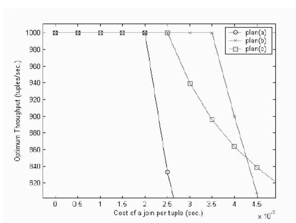




The plan with the lowest utilization is not always the best for load shedding

- **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 is 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) = 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)**

22

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 if 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.**

23

Outline

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

24

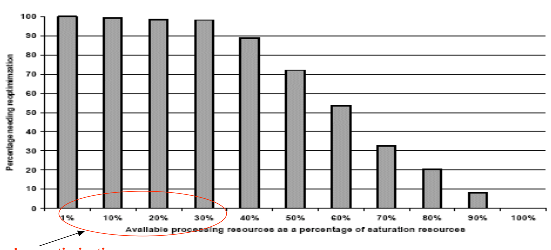
Experiments

- 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
-

$f_{A \bowtie B}$	0.2	W_A	100
$f_{A \bowtie C}$	0.5	W_B	300
$f_{B \bowtie D}$	0.1	W_C	500
$f_{D \bowtie E}$	0.001	W_D	400
		W_E	1000

25

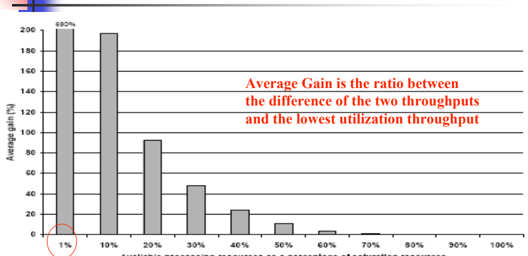
The Need for Reoptimization



Needs reoptimization

26

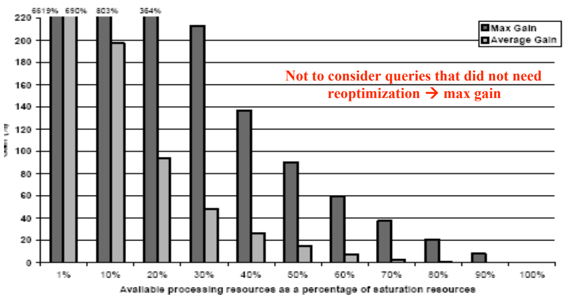
Experiments – Average Gain



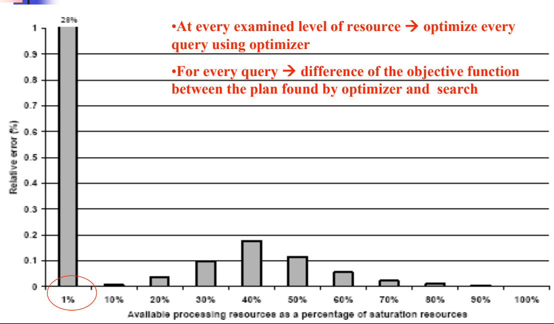
Low Resource,
Significant gain

27

Experiments – Max Gain



Experiments – Heuristic Optimizer



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**

30

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

31

QUESTIONS?

32