Query Processing

Jon Frankel, Noi Jencharat, Ened Ketri,
Anurag Maskey, Andy See, Larissa Smelkov

3/25/03

Opening Game - Who am I?

- Professor at the University of Wisconsin – Madison
- Specializing in database performance issues (i.e. joins)
- **Bonus**: What stream system have I worked on?

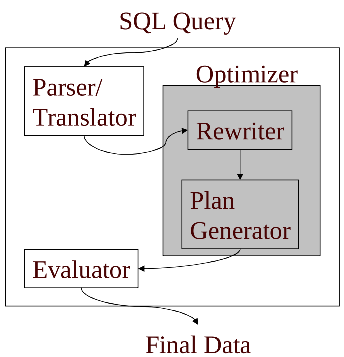
Query Processing – Papers

- Stratis Viglas and Jeffrey F. Naughton. **Rate-based query optimization for streaming information sources.** SIGMOD Conference 2002
- Jaewoo Kang, Jeffrey F. Naughton and Stratis D. Viglas. **Evaluating Window Joins Over Unbounded Streams.** VLDB 2002.
- S. Babu and J. Widom. **Exploiting k-Constraints to Reduce Memory Overhead in Continuous Queries over Data Streams.** Technical Report, Stanford University, November 2002.

Query Processing – Today's Agenda

- 1:40 Motivation & Setup Examples
- 2:20 Rate Based Query Paper
- 2:50 Break
- 3:00 Window Joins Paper
- 3:30 K-Constraints Paper
- 4:00 Discussion

127 Flashback



- Optimizer - cost based
- Select * from students
  where major = 'cosi'
  and birthday = '0325'

Stream Challenges

- Final Answer?
- Block Reads?
- Cardinality?

Stream Challenges

- ~~Final Answer?~~
- ~~Block Reads?~~
- ~~Cardinality?~~

Rate Based Cost Estimating

•Load Shedding
•Ad Hoc Queries
•Persistent Queries

Rate Based Analysis

MFC – 10/day; JDF 3/day

Day	New	MFC	Left	JDF	Left					
1	6	6 -> 3	0	3 -> 1	0					
2	9	9 -> 4	0	3 -> 1	1					
3	12	10 -> 5	2	3 -> 1	3					
4	0	2 -> 1	0	3 -> 1	1					
5				1 -> 0	0					
6										
7										
8										
9										

Rate Based Analysis

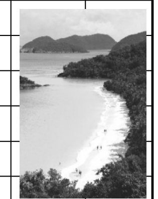
MFC – 10/day; JDF 3/day

Day	New	MFC	Left	JDF	Left		JDF	Left	MFC	Left
1	6	6 -> 3	0	3 -> 1	0		3 -> 1	3		
2	9	9 -> 4	0	3 -> 1	1		3 -> 1	9		
3	12	10 -> 5	2	3 -> 1	3		3 -> 1	18		
4	0	2 -> 1	0	3 -> 1	1		3 -> 1	15		
5				1 -> 0	0		3 -> 1	12		
6							3 -> 1	9		
7							3 -> 1	6		
8							3 -> 1	3		
9							3 -> 1	0		

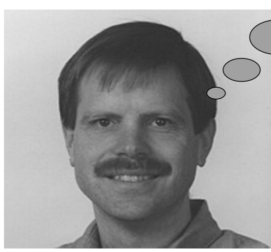
Rate Based Analysis

MFC – 10/day; JDF 3/day

Day	New	MFC	Left	JDF	Left		JDF	Left	MFC	Left
1	6	6 -> 3	0	3 -> 1	0		3 -> 1	3		
2	9	9 -> 4	0	3 -> 1	1		3 -> 1	9		
3	12	10 -> 5	2	3 -> 1	3		3 -> 1	18		
4	0	2 -> 1	0	3 -> 1	1		3 -> 1	15		
5				1 -> 0	0		3 -> 1	12		
6							3 -> 1	9		
7							3 -> 1	6		
8							3 -> 1	3		
9							3 -> 1	0	9 -> 4	0



Cost
Optimization
- Speed??



Coming Up….

-Different ways to measure rates
- SPJ applicability

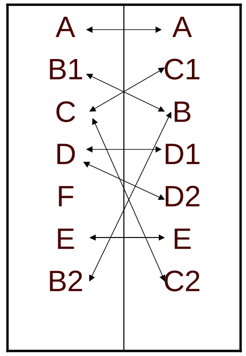
127 Flashback – Joins

- Predicate Pushdown
- Select * from students as s, courses as c
  where s.major = ‘cosi’
  and c.dept = ‘cosi’
  and s.sid = c.sid

Stream Challenges II

- Blocking Query Operators
  - (option: pipelined join)
- Lost/Delayed/Unordered Data
- And yet, benefits are

huge…



Stream Challenges II

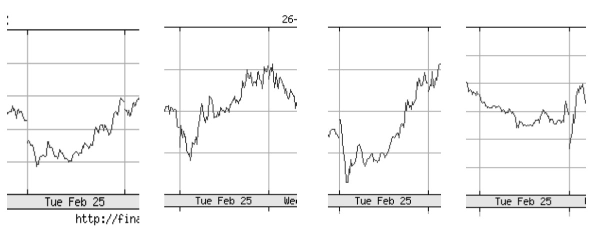
- Blocking Query Operators
  - (option: pipelined join)
- Lost/Delayed/Unordered Data
- And yet, benefits are

huge…

A	A
B1	
	C1
C	B
	D1
D	D2
F	E
E	
B2	C2

Stock Market – Econ 2A

Stock prices are based on ?



S&P Federated Dept Stores Home Depot Papa John's

Data is Out there!

(http://biz.yahoo.com/cc/)

Thu Mar 20 Times are U.S. Eastern

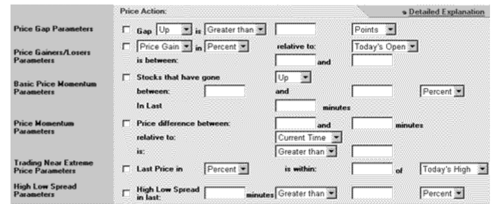
8:30 am CYCL Centennial Communications Earnings (Q3 2003)
8:30 am DV DeVry Inc. Acquires Ross University
8:30 am ENTG Entegris, Inc. Earnings (Q2 2003)
8:30 am PLXS Plexus Announcement
9:00 am HOLL Hollywood Media Corp. Fourth Quarter and Year-End 2002
9:00 am LEH Lehman Brothers Holdings First Quarter 2003 Earnings
10:00 am CSCO Cisco Systems Announces Agreement to Acquire The Linksys Group, Inc.
10:00 am FNLY Finlay Enterprises, Inc. Earnings (Q4 2002)
10:00 am GIII G-III Apparel Group Earnings (Q4 2003)
10:00 am GLYN Galyan's Trading Company, Inc. Fourth Quarter 2002
10:00 am MWD Morgan Stanley Earnings (Q1 2003)
10:00 am TRMS Trimeris, Inc. Earnings (Q4 2002)
10:30 am GPN Global Payments Inc. Earnings (Q3 2003)
11:00 am GDT Biosensor`s Agreement/Drug Eluting Stent Update
11:00 am CRAI Charles River Associates Earnings (Q1 2003)
11:00 am CHKR Checkers Drive-In Restaurants Earnings (Q4 2002)
11:00 am CPWM Cost Plus Earnings (Q4 2002)
11:00 am JCREW J. Crew Group, Inc. Earnings (Q4 2003)

Stocks & Stream Systems

Tickers	EPS (est)	EPS (actual)
1. IBM 80.00 1. INTC 15.00 1. HD 22.00 2. IBM 80.50 2. INTC 22.25 5. HD 22.75 9. HD 23.00	1. HD 0.27	 2. HD 0.30

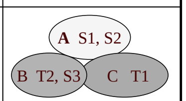
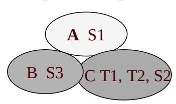
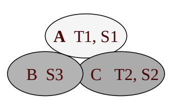
TRADERBOT®

Query: *Short-term Downward Momentum:*
Find all NASDAQ stocks between $20 and $200 that have moved down more than 2% in the last 20 minutes and there has been significant buying pressure (70% or more of the volume has traded toward the ask price) in the last 2 minutes.



Or by:
Earnings,
News,
Industry

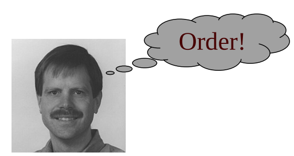
Aurora Example

Soldiers (time, ID, pos)	Tanks (time, ID, pos)	Problem?
1, S1, A 1, S2, A 2, S1, A	1, T1, C 2, T1, C	
1, S3, B 2, S3, B 3, S1, A	1, T2, B 2, T2, C	
3, S3, B 2, S2, C 3, S2, B	3, T1, A 3, T2, C	

Join Challenges – Window Options

- Aurora Option (by individual tuple)
- Stream Option (slide)

Tuple vs Timestamp

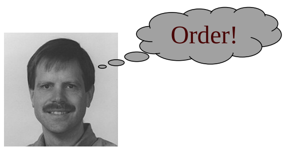


A	A
B1	C1
C	B
D	D1
F	D2
E	E
B2	C2

Join Challenges – Window Options

- Aurora Option (by individual tuple)
- Stream Option (slide)

Tuple vs Timestamp



A	A
B1	C1
C	B
D	D1
F	D2
E	E
B2	C2

Join Challenges – Window Options

- Aurora Option (by individual tuple)
- Stream Option (slide)

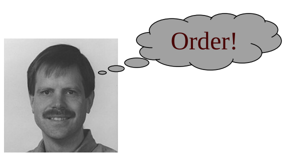
Tuple vs Timestamp

A	A
B1	C1
C	B
D	D1
F	D2
E	E
B2	C2

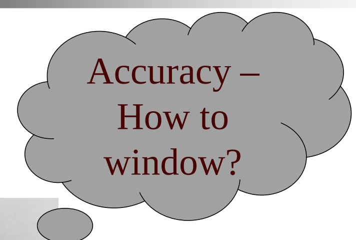
Join Challenges – Window Options

- Aurora Option (by individual tuple)
- Stream Option (slide)

Tuple vs Timestamp



A	A
B1	C1
C	B
D	D1
F	D2
E	E
B2	C2



Coming Up….

-Joining algorithms
-Lots of cool graphs

Motivations

- Traditional Optimizers requires cardinality of the input….
- In streams, cardinality is not known and inputs come at different rate…
- RATE-BASED optimization

What is Rate?

- Number of records per a unit of time.
- Output Rate =

$$\frac{\text{\# output transmitted}}{\text{time needed for transmission}}$$

$$\text{Output Rate} = \frac{\text{\#papers}}{\text{processing time needed}}$$

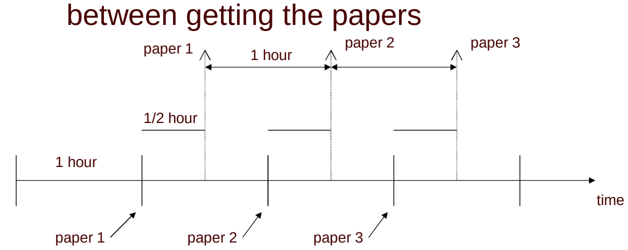
Output Rate Estimation

- For Projections
- For Selections
- For Joins

Output Rate for Projections

- case 1: Mitch

Time to read papers is shorter than time between getting the papers

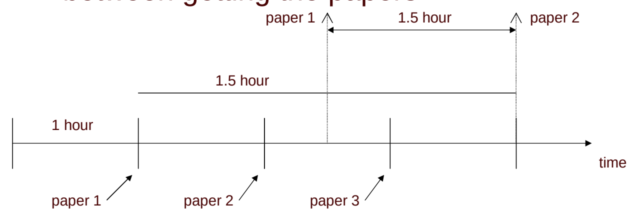


So the output rate = the input rate

Output Rate for Projections

- case 2: Jon

  Time to read papers is longer than time between getting the papers



So the output rate = 1/(time to do projection)

Output Rate for Projections

- In general, time to do projection is low.
- So

Output Rate = Input Rate

r_o r_i

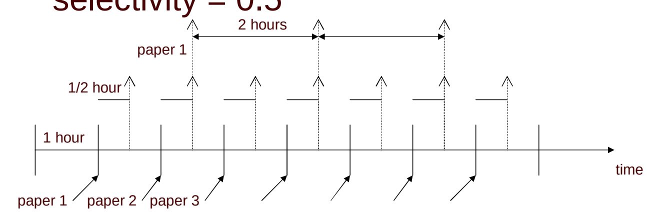
Output Rate for Selections

- Selectivity (f) = percentage of papers that will be selected

Output Rate for Selection

- case 1: Mitch

  takes 1/2 hour to read 1 paper, with selectivity = 0.5

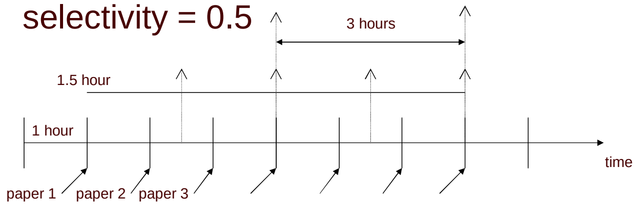


output rate = 1/2 paper/hour

So the output rate = f * the input rate

Output Rate for Selection

- case 2: John
  takes 1.5 hour to read 1 paper, with selectivity = 0.5



output rate = 1/3 paper/hour (= 1/2 * 1/1.5)
So the output rate = f * (1/time to select)

Output Rate for Selections

- In general, time to perform selection is less than interval between inputs.
- So

Output Rate = Selectivity * Input Rate

$$r_o = f * r_i$$

Output Rate for Joins

- What are the papers by same author Mitch and Jon gives the same grading to?
- r_M = No. of papers Mitch reads per hour
- r_J = No. of papers Jon reads per hour
- f = Selectivity of join
- C_M = Time to handle reviews from Mitch
- C_J = Time to handle reviews from Jon

Recall

- Output Rate =

Total #'s of papers in output

$$\frac{\text{\# output transmitted}}{\text{time needed for transmission}}$$

Total time to do the Join

Output Tuples

- time interval = t:

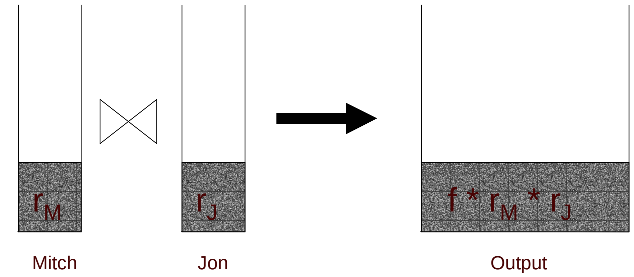
  We have:

  r_M*t paper reviews from Mitch

  r_J*t paper reviews from Jon

  $f*r_M*r_J*t^2$ tuples that can be in the output.

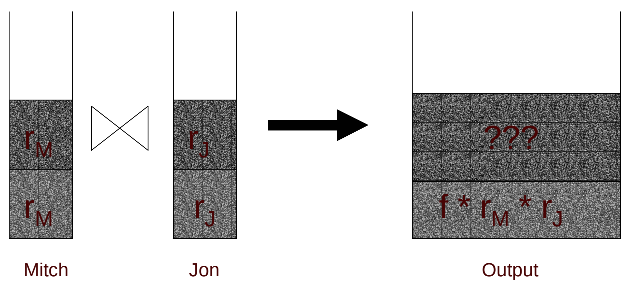
after 1 hour



Number of output tuples:

$f * r_M * r_J$

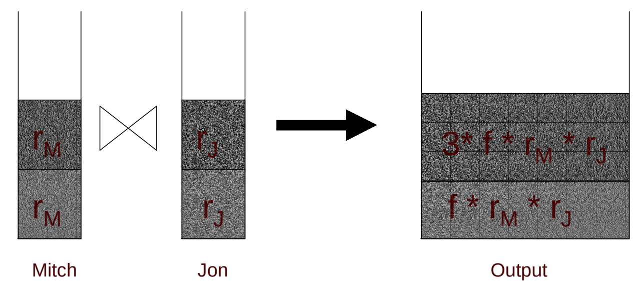
after 2 hours



Number of output tuples:

$f * r_M * r_J \quad + f*r_M*2r_J \quad + f*2r_M*r_J \quad - f*r_M*r_J$

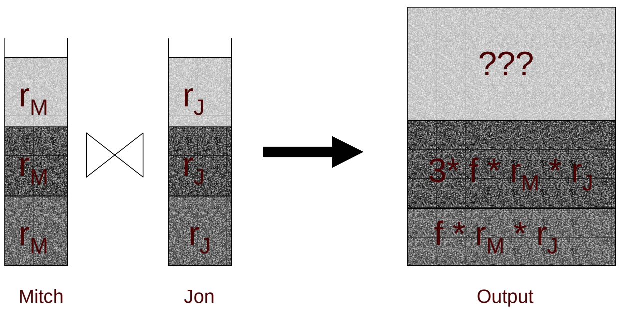
after 2 hours



Number of output tuples:

$f * r_M * r_J \quad + 3 * f * r_M * r_J$

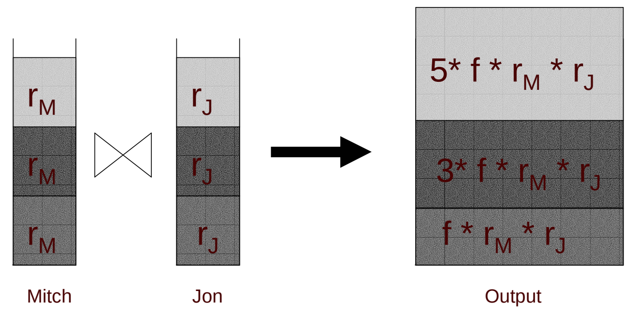
after 3 hours



Number of output tuples:

$f * r_M * r_J + 3 * f *$ rM * rJ $+ f*r_M*3r_J + f*3r_M*r_J - f*r_M*r_J$

after 3 hours



Number of output tuples:

$f * r_M * r_J + 3 * f * r_M * r_J + 5* f * r_M * r_J$

after time t

- There will be (2t-1) $f* r_M * r_J$ new tuples in the output.
- Total number of outputs at time t:
- $\int ((2t-1)\, f* r_M * r_J)\, dt$

  $= t^2 * f* r_M * r_J – t* f* r_M * r_J$

  $= f* r_M * r_J * t *(t-1)$

Time to Process Join

- at time t

	Inputs	Time	Processing Time
Mitch	r_M*t	C_M	r_M*t * C_M
Jon	r_J*t	C_J	r_J *t * C_J

- Total time = r_M*t * C_M + r_J*t * C_J

  = t(r_M*C_M + r_J* C_J)

Output Rate for Joins

$$\frac{\#\text{ output transmitted}}{\text{time needed for transmission}}$$

$$= \frac{f^* r_M * r_J * t *(t-1)}{t(r_M*C_M + r_J* C_J)} = \frac{f^* r_M * r_J *(t-1)}{(r_M*C_M + r_J* C_J)}$$

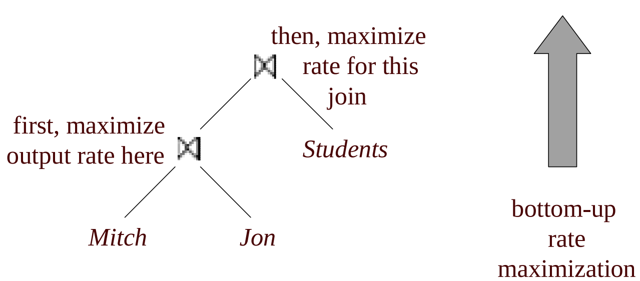
$$\approx \frac{f^* r_M * r_J * t}{r_M*C_M + r_J* C_J}$$

Optimizing Queries

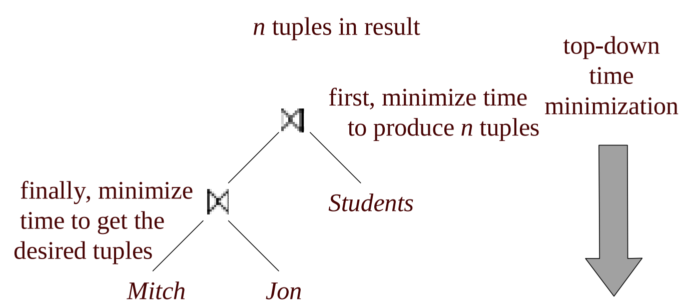
$$\#\text{ outputs} = \int_0^p r(t)\,dt$$

- Optimize for a specific time point
  - *which plan will produce the most results by t_o?*
- Optimize for output production size
  - *which plan is the first one to reach N results?*

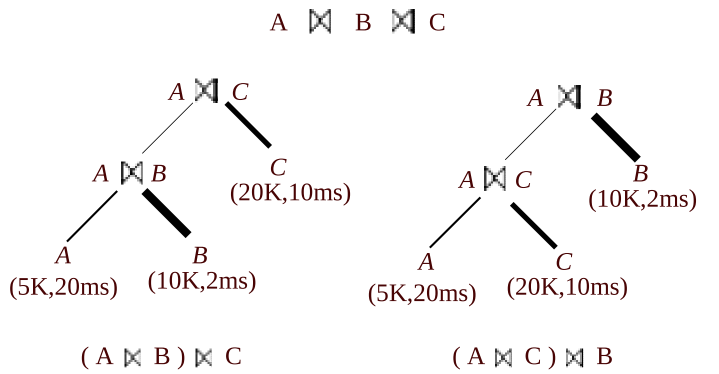
Local Rate Maximization



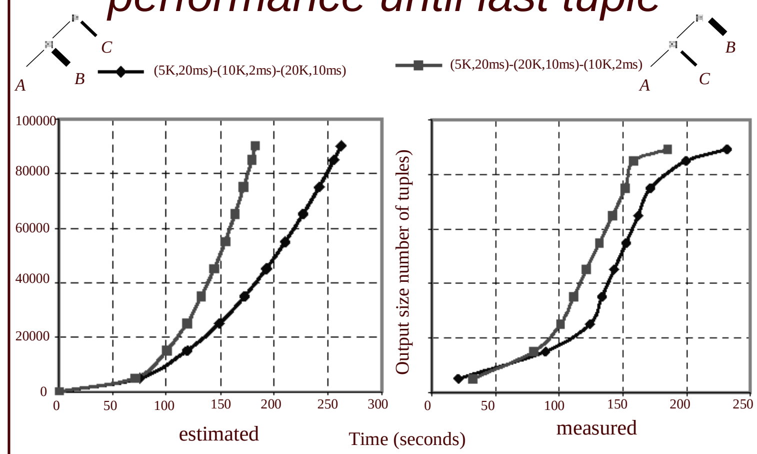
Local Time Minimization



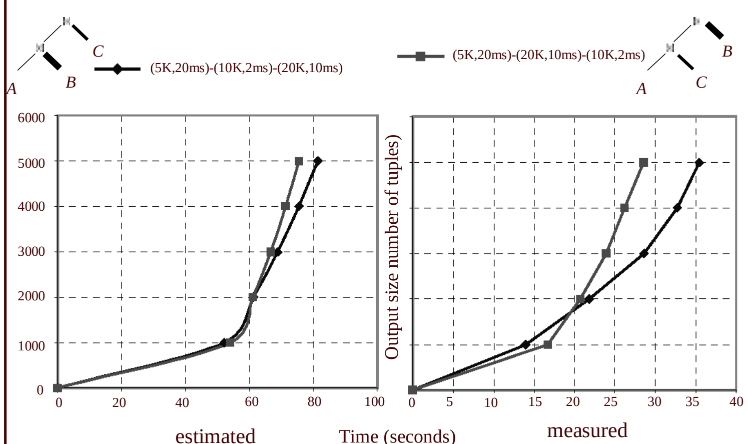
Experiment I
the plans



Experiment I
performance until last tuple

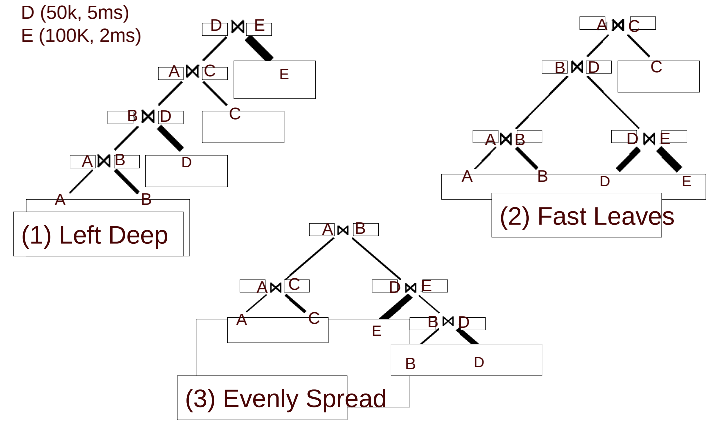


Experiment I
performance for the first few thousand tuples

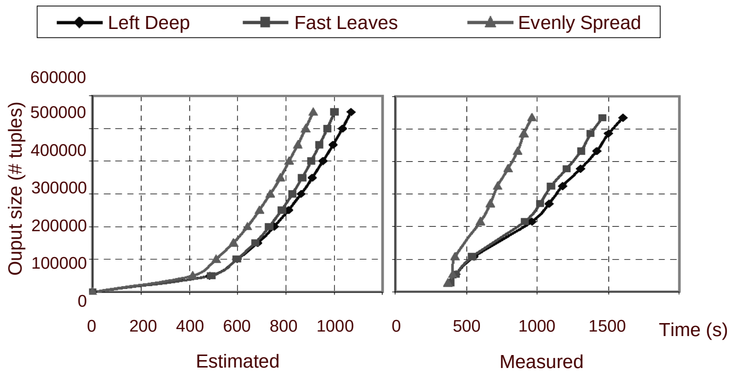


A (5K, 20ms)
B (10K, 10ms)
C (20K, 15ms)
D (50k, 5ms)
E (100K, 2ms)

Complex Plans



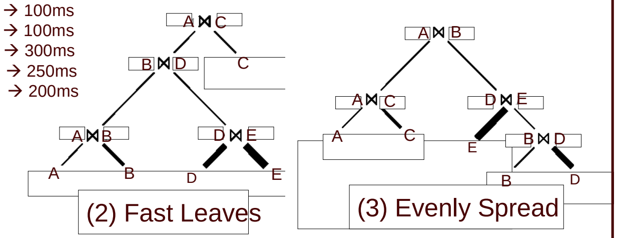
Experiment Result



Comparison to Traditional Model

Plan	Traditional Est.	Rate-Based Est.
Left Deep	10^4	$1.3 * 10^3$
Fast Leaves	$2 * 10^3$	$9.7 * 10^2$
Evenly Spread	$5 * 10^3$	$8.8 * 10^2$

A (5K, 20ms) → 100ms
B (10K, 10ms) → 100ms
C (20K, 15ms) → 300ms
D (50k, 5ms) → 250ms
E (100K, 2ms) → 200ms

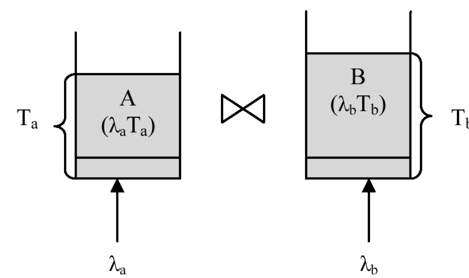


Evaluating Window Joins over Unbounded Streams

- Jaewoo Kang
- Jeffrey F. Naughton
- Stratis D. Viglas

University of Wisconsin-Madison
Computer Sciences Department

Moving Window Join

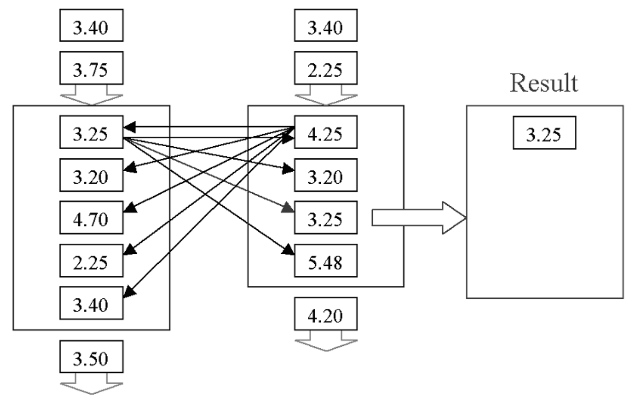


T_b - stream B time window size λ_b - stream B arrival rate

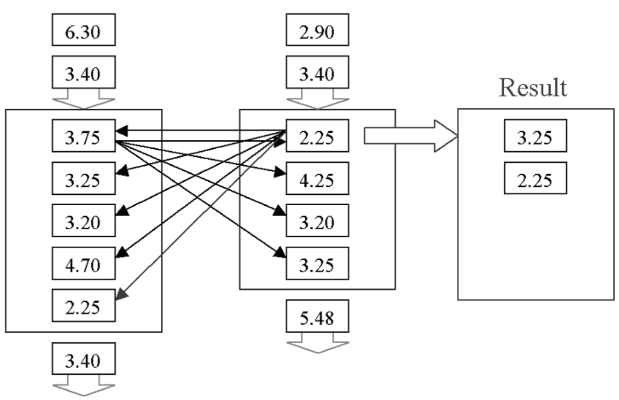
Types of Join

- Nested loops join
- Hash join

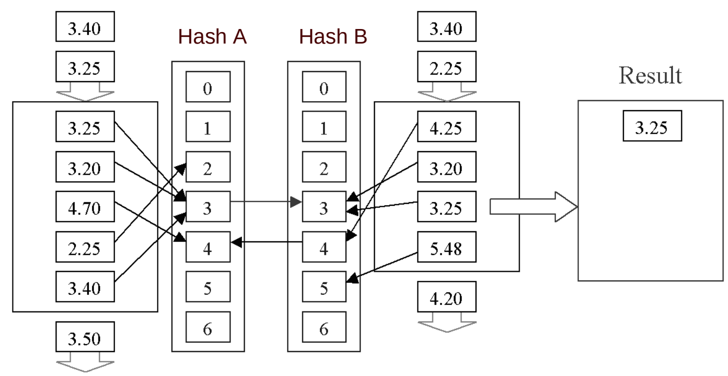
Nested Loops Join



Nested Loops Join



Hash Join



Open Questions

- How to measure the efficiency of a moving window join?
- Can the join of streams with different rates be more efficient?
- How to deal with fast input streams when system cannot manage them?
- How to share limited memory between the two windows for the two inputs?

Cost of Moving Window Joins (unit-time basis model)

$$C_{A\bowtie B} = \lambda_a(probe(b) + insert(a) + invalidate(a)) + \lambda_b(probe(a) + insert(b) + invalidate(b))$$

$$\Downarrow$$

$$C_{A\bowtie B} = \lambda_a(probe(b)) + \lambda_b(insert(b) + invalidate(b)) + \lambda_b(probe(a)) + \lambda_a(insert(a) + invalidate(a))$$

Idea!

Streaming join algorithms can be asymmetric

Hash – Nested Loops join

Nested Loops – Hash join

… or symmetric

Nested Loops – Nested Loops join

Hash – Hash join

Cost of Join

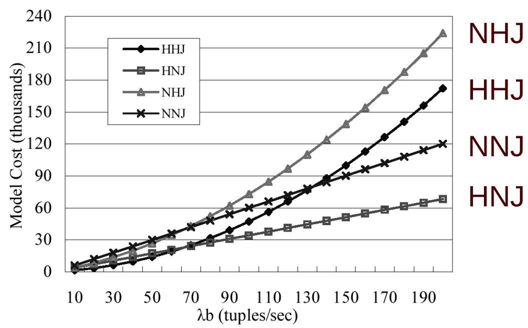
- Nested loops join

$$C_{A\bowtie B}(NLJ) = (\lambda_a T_b \lambda_b + 2\lambda_b) \times C_n + (\lambda_b T_a \lambda_a + 2\lambda_a) \times C_n$$

- Hash join

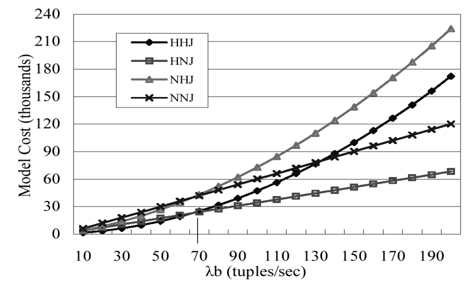
$$C_{A\bowtie B}(HJ) = (\lambda_a T_b \lambda_b \sigma_b + \lambda_b T_b \lambda_b \sigma_b + \lambda_b) \times C_h + (\lambda_b T_a \lambda_a \sigma_a + \lambda_a T_a \lambda_a \sigma_a + \lambda_a) \times C_h$$

Comparison of Joins



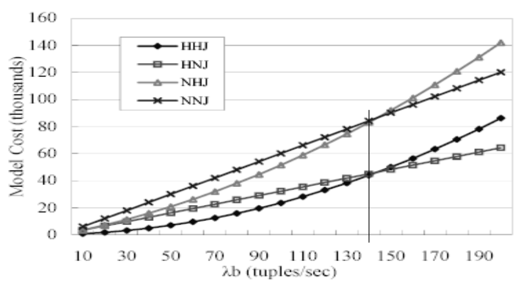
T_a=60, λ_a=10, T_b=60, σ_a=0.1, σ_b=0.1, C_n=0.5, C_h=0.65

Full Joins



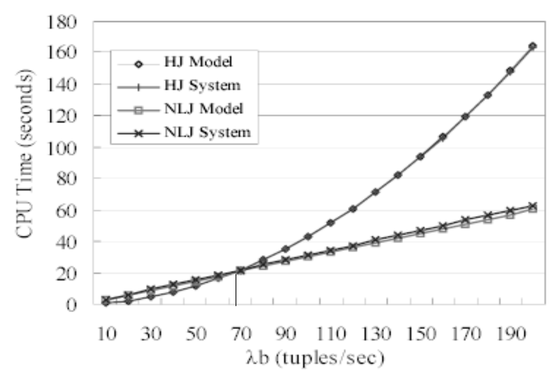
T_a=60, λ_a=10, T_b=60, σ_a=0.1, σ_b=0.1, C_n=0.5, C_h=0.65

Full Joins: different selectivity



σ_a=0.05, σ_b=0.05.

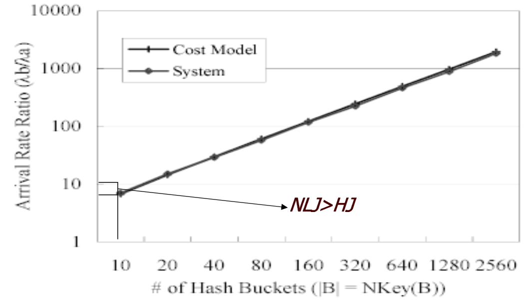
1-way Join: System/Model cost



Overhead Costs

- ***C_h/C_n***
  - Ratio of overhead cost of Hash Join to Nested Loop join
  - Model: ratio = 1.3
- |B|
  - Number of hash buckets in window B, assumed same as number of unique keys in window B
  - Variable that can be changed in the model

Crossover Points



$$\frac{\lambda_b}{\lambda_a} \approx \frac{|B| - C_h/C_n}{C_h/C_n}$$

Output rates

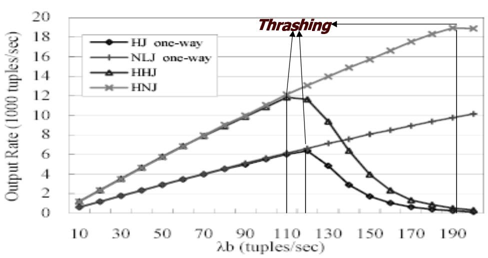


Figure 7. Join output rates with varying λ_b (T_a=60, T_b=60, λ_a=10, σ_a=0.1, σ_b=0.1)

$$\frac{\lambda_b}{\lambda_a} > \frac{|B| - C_h/C_n}{C_h/C_n}$$

CPU time

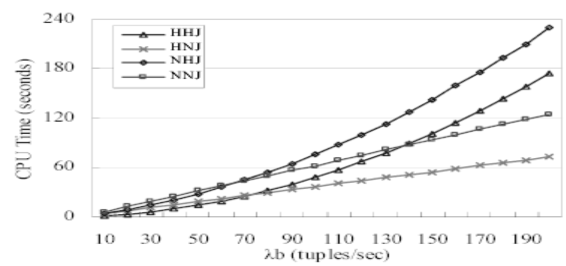


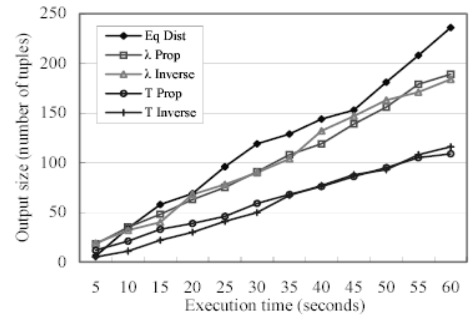
Figure 6. CPU Time of four full joins (T_a=60, T_b=60, λ_a=10, σ_a=0.1, σ_b=0.1)

Insufficient Resources for handling the Stream Input Rates

- Problem
  - Very Expensive Predicates
  - Input rate > Join operator service rate

- Solution
  - Drop tuples from input

Resource Allocation Strategies



λ_a=80, λ_b=20, T_a=9, T_b=1 μ=10

Limited Memory

- Variable Time Window

- Allocate Memory depending on Stream Rate

Memory Allocation Strategies

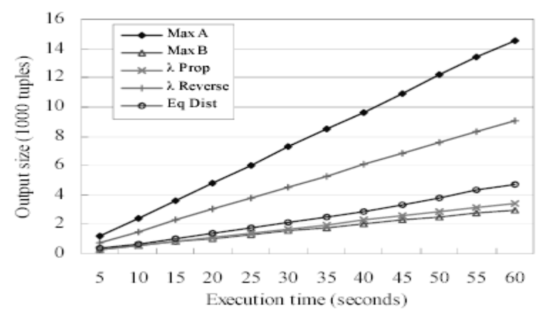


Figure 9. Performance of memory allocation strategies w/ fixed arrival rates (λ_a=10, λ_b=50, M=1000, σ_a=0.005, σ_b=0.01)

Memory Allocation Strategies

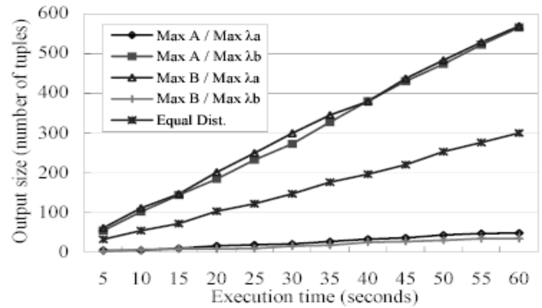


Figure 10. Performance of memory allocation strategies w/ variable arrival rates (μ=10, M=100, σ=0.01)

Memory Allocation Implications

- Give all memory (biggest window size) to slowest input stream
  - Fast stream probes slow stream, skips insertion/invalidation
  - Full Join reduces to One Way Join on the direction of slow → fast
  - Choose Join Algorithm after memory allocation

Conclusions

- A Full Join can be seen as two separate independent Single Joins
  - Exploit asymmetrical stream input rates
    - NLJ/HJ algorithms Combination
      - HNJ/NHJ best candidate
    - Resource allocation
      - Devote most resources to slowest stream

K-Constraints

Exploiting k-Constraints to Reduce Memory Overhead in Continuous Queries over Data Streams

Shivnath Babu and Jennifer Widom, Stanford University

Introduction

- Already saw:
  - Use **Rate** information to optimize.
- Now we'll see
  - Use properties of streamed data.
  - In order to reduce memory usage.

Outline

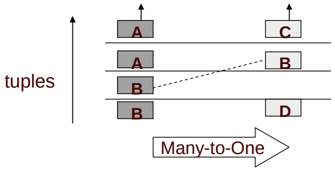
- Constraints for streams
- K-constraints
- Synopsis
- Algorithm using k-constraints

Constraints

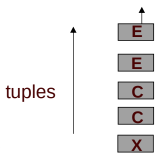
- Properties that data streams satisfy.
- Examples:
  - *Many-one* join constraints between two streams.
  - *Referential-integrity* constraints for streams
    - Between two streams in many-one join
    - "One" side arrives *before* "Many" side
  - *Clustered-arrival* constraints on an attribute
    - *Duplicate* values arrive *together*
  - *Ordered-arrival* constraints on an attribute
    - Values are *clustered* and *ordered*.

Constraints (visual)

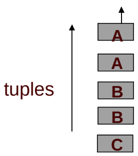
- Referential Integrity



- Clustered Arrival



- Ordered Arrival



Constraints ?

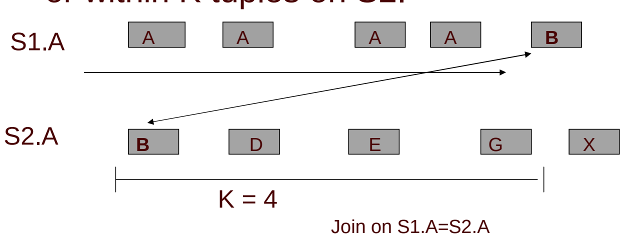
- How practical are these constraints for streams?
- Tuples may come out of order.
  - Clustered? Ordered?
- Data rate may vary.
  - Referential Integrity?

K-Constraints

- Idea: allow some disorder.
- K-Constraints are:
  - Constraints that are almost met.
  - K is the adherence parameter
    - Lower K means streams comes closer to the constraint.
  - Like “slack” in Aurora
    - Set amount of disorder can be tolerated by system.
  - Examples:

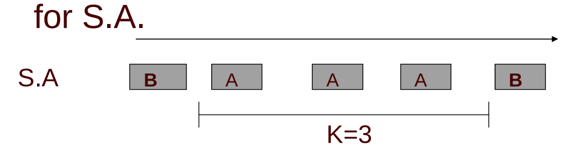
Referential Integrity

- Many-one join from S1 to S2.
- S2 tuple will arrive before joining S1 tuple, or within K tuples on S2.



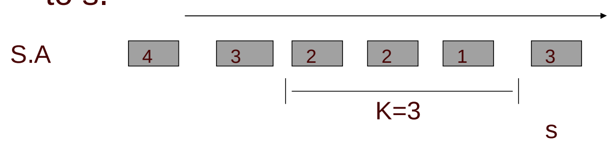
Clustered arrival

- On attribute S.A:
- At most k tuples with different S.A values arrive between tuples with the same value for S.A.



Ordered arrival

- On stream attribute S.A:
- Tuples that arrive at least k+1 tuples after tuple s have a value greater than or equal to s.



The Idea

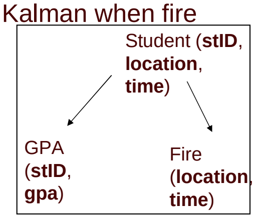
- Joins over streams take infinite memory.
- Idea is to use k-constraints to reduce memory usage
  - Slower increase in memory usage.
  - Constant memory usage in some cases.
- K-constraints can decide which tuples to keep around.

Terminology

- **Synopsis**: stream history
- Each Synopsis for a stream involved with a query:
  - Has 3 components of seen tuples:
    - *Yes*: may contribute to a result tuple
    - *No*: cannot contribute to a result tuple
    - *Unknown*: cannot be put in Yes or No.
- **Join Graph**: directed graph with arcs from "Many" (parent) to the "One" (child) of many-one join.

Synopsis example

Query: Students that have GPA < 3.0 in Kalman when fire alarm is on.



Stream Student gets tuple: (**id1234**, **Kalman**, **12:05**)

Stream Fire gets tuple: (**id1234**, **Kalman**, **12:00**)

(**Kalman**, **12:05**) (**Edison**, **12:00**)

OUTPUT: (**id1234**, **2.9**, **Kalman**, **12:05**)

	Yes	No	Unknown
Student			
GPA	(**id1234**, **2.9**)		
Fire			

Synopsis

- Why not just keep those tuples that are in the *Yes* or *Unknown* synopsis?
- Might cause tuples in other streams to be kept in *Unknown* rather than being discarded.

Synopsis example 2

Soldiers with heartrate **= 0** where more than **2** missiles were seen.

Stream Heart (SoldierID,Rate) gets tuple: (s1,**1**) (s2,**0**)

OUTPUT

Stream Missile(Sector) gets tuple: (**Sec3**, **1**) (**Sec5**, **4**) (s2,0,**Sec5**)

	Yes	No	Unknown
Heart(soldID, Rate)			(s3,**0**)
Where(soldID,Sector)			(s2,**Sec5**) (s3,**Sec3**)
Missiles(Sector)			

Heart (ID,**Rate**)

Where (ID,**Sector**)

Missile (**Sector**, **Number**)

Referential Integrity

Join heart rates greater than **35** with soldiers in sector 3 on id and time.

Constraints:
- location gets transmitted first
- always arrives within 2 tuples of heart rate.

loc = sec3

id	time	loc
s3	0	**sec2**
s2	1	**sec2**
s1	2	**sec3**
s3	1	**sec3**

many-one

rate > 35

id	time	rate
s1	1	**39**
s1	1	**28**
s3	1	**38**

Many

One

OUTPUT: (s3,1,**sec3**,**38**)

YES	NO	UNK
s1,2,**sec3**	s3,0,**sec2**	
s3,1,**sec3**	s2,1,**sec2**	

YES	NO	UNK
	s1,1,**28**	s1,1,**39**
		s3,1,**38**

Since more than 2 tuples have come on left, this can be moved to No.

The *No* synopsis on left is never needed!!!

Neither is *No* on the right.

Referential Integrity

- If Referential Integrity with parameter K holds on many-one join S1 to S2
  - Eliminate S2's *No* component
  - Keep S1's *Unknown* component for only k tuples on S2.

HeartRate

Location

Ordered-Arrival Constraints (OA(k))

Two algorithms:

- On child stream ("one" in many-one join)
  - OAC(k)
- On parent stream ("many" in many-one join)
  - OAP(k)

Ordered Arrival (on "one")

Soldiers in **sector 3** while a soldier had heart rate of **0**. (Join on time. Assume one location tuple per time.)
Constraint:
•**Location** comes in ordered, with at most 1 tuples out of order.

loc = sec3

id	time	loc
s4	1	**sec2**
s5	2	**sec2**
s7	4	**sec3**

many-one

rate =0

id	time	rate
s2	1	**0**
s1	4	**0**
s3	0	**38**

OUTPUT: Many One (s7,s1, 2,**sec3**,**0**)

YES	NO	UNK
s7,4,**sec3**	s4, 1,**sec2**	
	s5,2,**sec2**	

YES	NO	UNK
		s2,1,**0**
	s3,0,**38**	s1,4,**0**

Since minimum on left will now be 2, we can move this to *No*!!!

The *No* synopsis on left is never needed!!!

Ordered Arrival (on "many")

Soldiers in **sector 3** while a soldier had heart rate of **0**. (Join on time. Assume one location tuple per time.)
Constraint:
•**HeartRate** comes in ordered, with at most 1 tuples out of order.

loc = sec3

id	time	loc
s4	0	**sec3**
s5	5	**sec2**
s7	2	**sec3**

many-one

rate =0

id	time	rate
s2	1	**0**
s1	2	**0**
s3	2	**38**

OUTPUT: Many One (s7,s1, 2,**sec3**,**0**)

YES	NO	UNK
s4, 0,**sec3**	s5,5,**sec2**	
s7,2,**sec3**		

YES	NO	UNK
	s3,2,**38**	s2,1,**0**
		s1,2,**0**

Since k+1 tuples with time > 0 have come on right, we can discard this!!

The *No* synopsis on left is never needed!!!

OAC(k)

- Similar to Referential Integrity
- Eliminate *No* synopsis without filling parent *Unknown* synopsis:
- Maintain the minimum value L that will be seen on stream S.
- Tuples in parent Stream less than L that do not match S's *Unknown* or *Yes*, must have no matching tuple in S – no need to put into *Unknown.*

OAP(k)

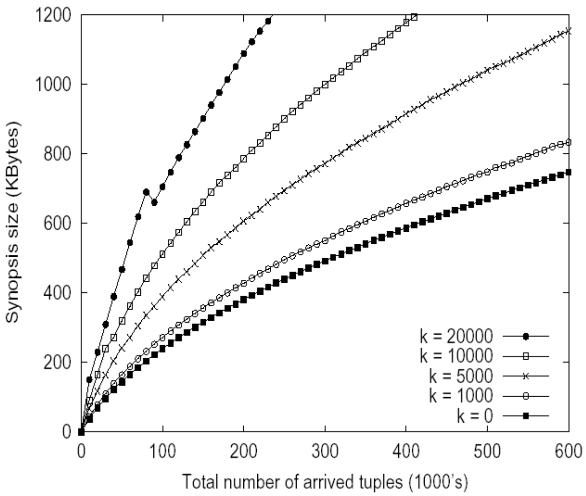
- Idea:
  - Given a child stream's tuple s,
  - If no future parent tuples can join with s,
  - Then, Don't store s.
- If Ordered Arrival constraint on parent stream's attribute A OAP(k)
  - Can drop child's tuples after k tuples with larger A values.

Clustered Arrival (CA(k))

- Idea:
  - Similar to Ordered arrival on parent stream.
- If parent streams have CA(k) on attribute A:
  - After a joining tuple in parent, store s for only k more parent tuples.

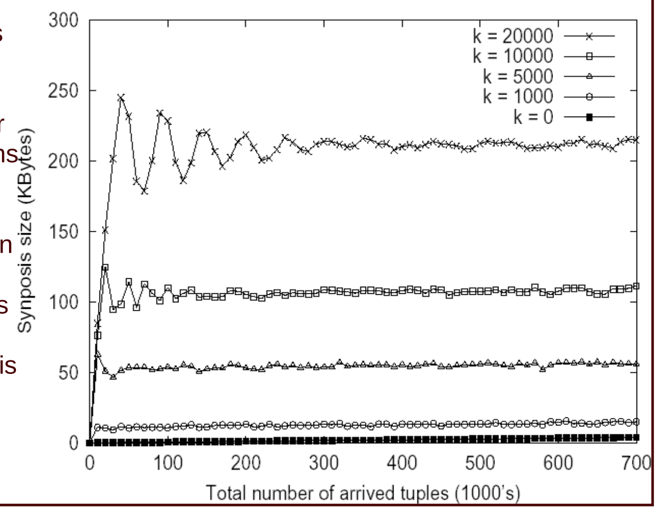
RIDS(k) Results

Larger k means tuples are kept in *Unknown* synopsis longer, using more memory.



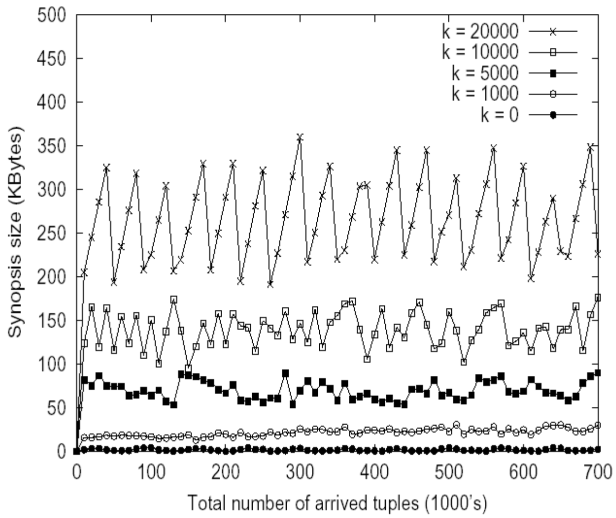
CA(k) Results

Smaller K means store fewer tuples in child streams *Yes* synopsis



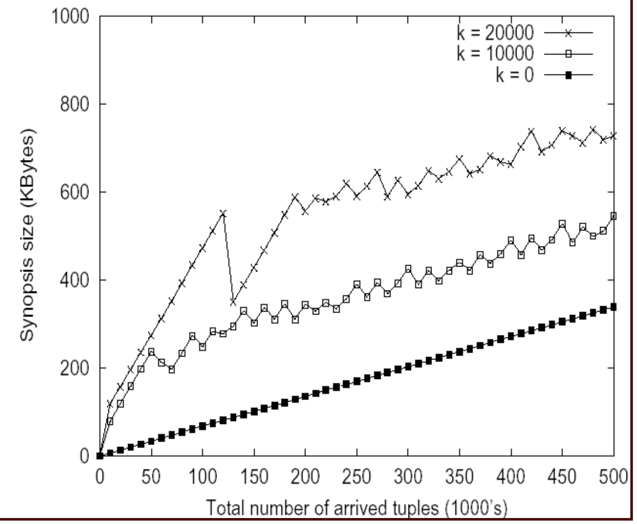
OAP(k) Results

Smaller K means store less in child *Yes* synopsis



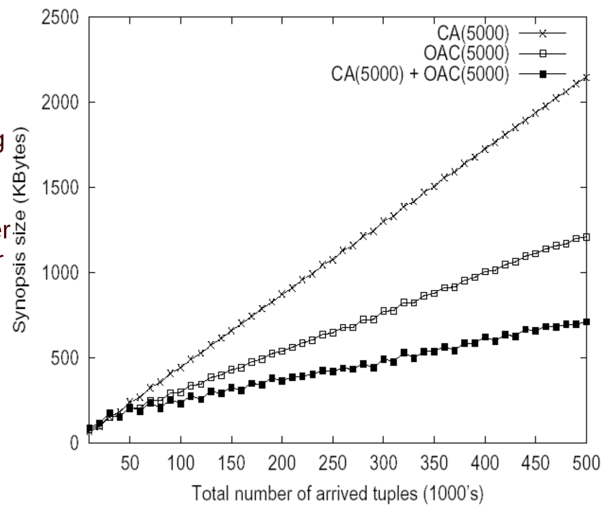
OAC(k) Results

Smaller K means tuples are kept in parent stream synopsis less time.



CA(k) and OAC(k)

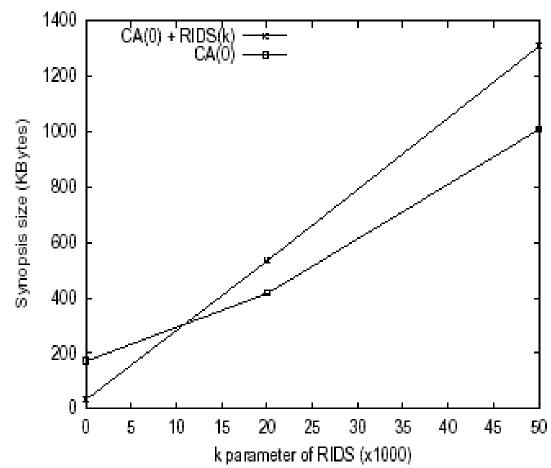
Combining CA(k) and OAC(k) does better than either alone, especially at high values for K.



CA(k) vs. combined CA(k) and RIDS(k)

Note that at low K for RIDS(k), CA(k) does better.

Some tuples are kept around longer than in pure CA(k).



Summary

Speed

Accuracy

- Cost Optimization
  Cardinality -> Rate
- Pin Slow Streams

- Windows for approximation
- Memory issues
- Join algorithms

Summary

Speed

Accuracy

- Cost Optimization
  Cardinality -> Rate
- Pin Slow Streams

- Windows for join approximation
- Memory issues
- Join algorithms

Discussion

- When join by timestamp with a range, what is timestamp of output tuple?
- How are punctuation and K-constraints similar?
- Rate based paper didn't account for windows – what is the effect?

Discussion

- What are the pros/cons of windows vs K-Constraints?
- The join paper assumed finite streams – do their conclusions work for infinite streams?
- Can you think of other cost measuring methods for the optimizer?

Discussion

- How would a stream system optimize across multiple, concurrent persistent queries? Does what we studied today apply?
- How would a stream system handle non-equijoins? Does what we studied today apply?

Open Questions

- Could this approach be used on systems like Aurora/Stream etc. ?
- Can this model be modified so that it can be applied to other operators, and if so, would it have good benefits?
- How much asymmetry actually exists in practice?