

Content-Based Routing: Different Plans for Different Data

Pedro Bizarro, Shivnath Babu, David DeWitt, Jennifer Widom

VLDB 2005

CS 632 Seminar Presentation

Saju Dominic

Feb 7, 2006

Introduction

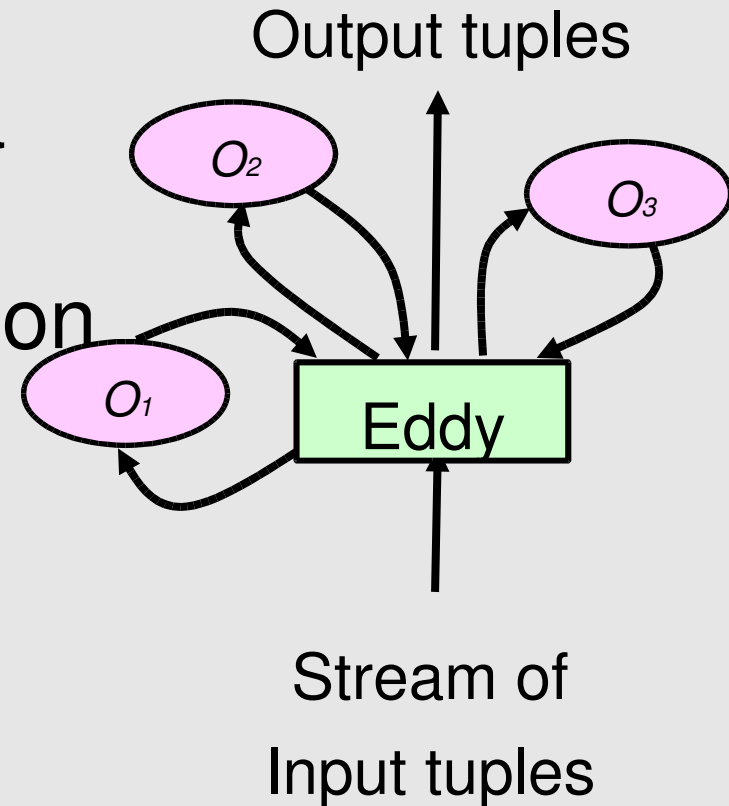
- Different parts of the same data may have different statistical properties.
- Different query plans may be optimal for the different parts of the data for the same query.
- Concurrently run different optimal query plans on different parts of the data for the same query

Overview of CBR

- Eliminates single plan assumption
- Identifies tuple classes
- Uses multiple plans, each customized for a different tuple class
- Adaptive and low overhead algorithm
- CBR applies to any streaming data:
 - stream systems
 - regular DBMS operators using iterators
 - and acquisitional systems.
- Implemented in TelegraphCQ as an extension to Eddies

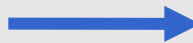
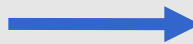
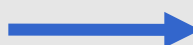
Overview of Eddies

- Eddy routes tuples in a particular order through a pool of operators
- Routing decisions based on operator characteristics:
 - Selectivity
 - Cost
 - Queue size



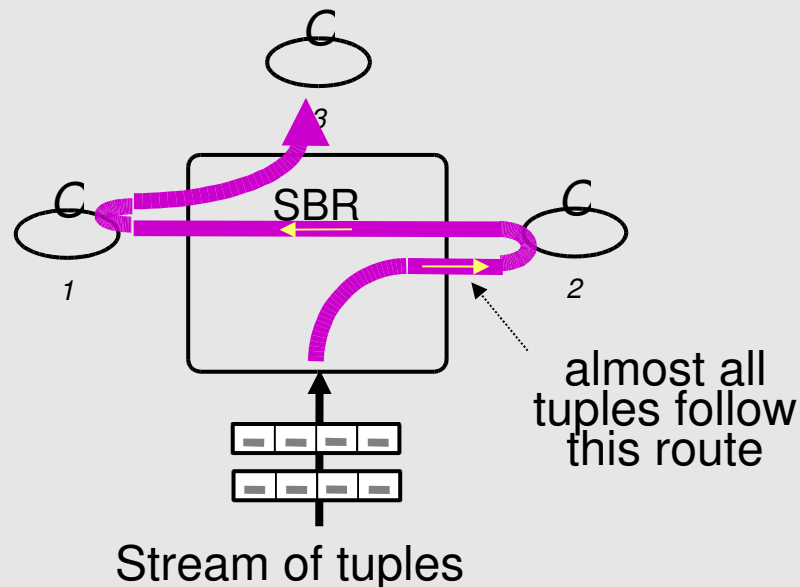
- Routing decisions not based on tuple content

Intrusion Detection Query

- “Track packets with destination address matching a prefix in table T, and containing the 100-byte and 256-byte sequences “0xa...8” and “0x7...b” respectively as subsequence”
- `SELECT * FROM packets`  O_1
`WHERE matches(destination, T)`  O_2
`AND contains(data, “0xa...8”)`  O_3
`AND contains(data, “0x7...b”);`

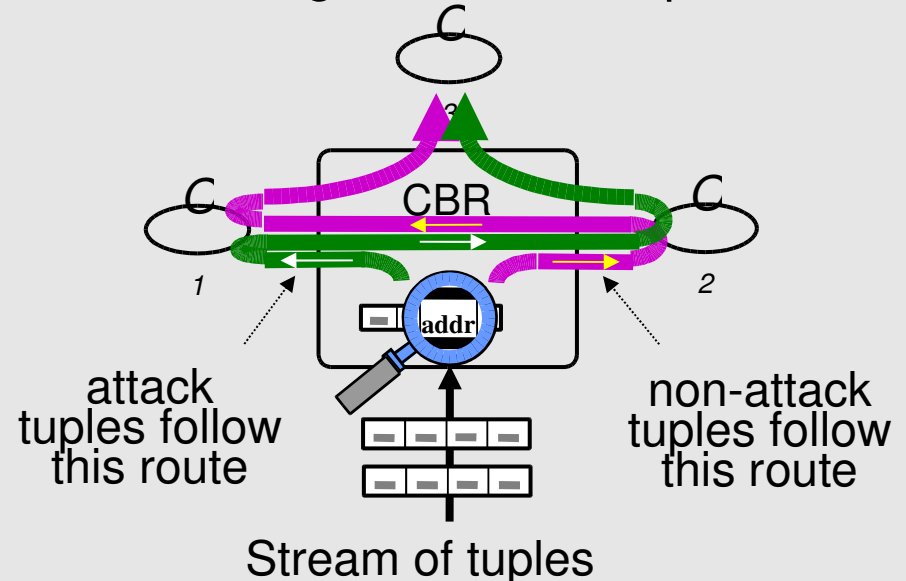
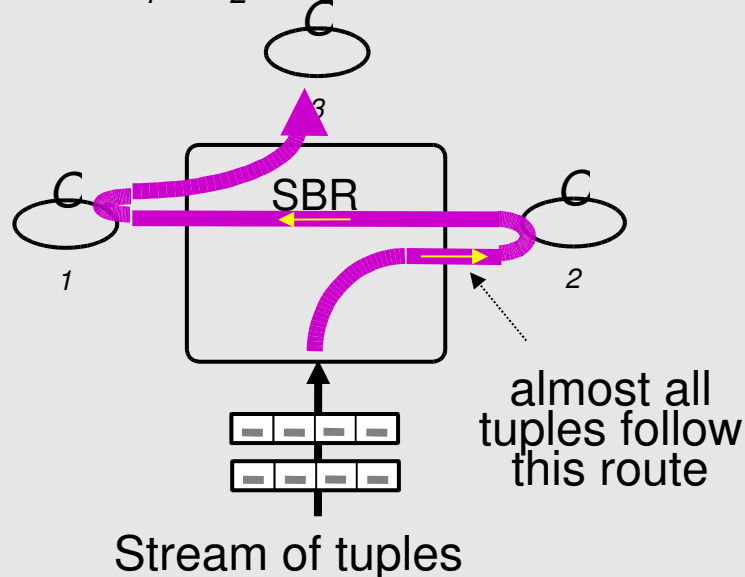
Intrusion Detection Query

- Assume:
 - costs are: $c_3 > c_1 > c_2$
 - selectivities are: $\sigma_3 > \sigma_1 > \sigma_2$
- SBR routing converges to O_2, O_1, O_3



Intrusion Detection Query

- Suppose an attack (O_2 and O_3) on a network whose prefix is not in $T(O_1)$ is underway:
 - O_2 and O_3 will be very high, O_1 will be very low
 - O_1, O_2, O_3 will be the most efficient ordering for “attack” tuples



Content-Based Routing Example

- Consider stream S processed by O_1 , O_2 , O_3

	O_1	O_2	O_3
Selectivities	30%	40%	60%

Overall Operator Selectivities

- Best routing order is O_1 , then O_2 , then O_3

Content-Based Routing Example

- Let A be an attribute with domain $\{a,b,c\}$

Value of A	O_1	O_2	O_3
$A=a$	32%	10%	55%
$A=b$	31%	20%	65%
$A=c$	27%	90%	60%
Overall	30%	40%	60%

Content-Specific Selectivities

- Best routing order for $A=a$: O_2, O_1, O_3
- Best routing order for $A=b$: O_2, O_1, O_3
- Best routing order for $A=c$: O_1, O_3, O_2

Classifier Attributes

- Goal: identify tuple classes
 - Each with a different optimal operator ordering
- CBR considers:
 - Tuple classes distinguished by content, i.e., attribute values
- Classifier attribute (informal definition):
 - Attribute A is classifier attribute for operator O if the value of A is correlated with selectivity of O .

Best Classifier Attribute Example:

- Attribute A with domain {a, b, c}
- Attribute B with domain {x, y, z}
- Which is the best to use for routing decisions?
- Similar to AI problem: classifier attributes for decision trees
- AI solution: Use GainRatio to pick best classifier attribute

		+			-
A=a	10%	90%	B=x	43%	57%
A=b	20%	80%	B=y	38%	62%
A=c	90%	10%	B=z	39%	61%
Overall	40%	60%	Overall	40%	60%

GainRatio to Measure Correlation

		f			f
A=a	10%	90%	B=x	43%	57%
A=b	20%	80%	B=y	38%	62%
A=c	90%	10%	B=z	39%	61%
Overall	40%	60%	Overall	40%	60%

$$\text{GainRatio}(R, A) = 0.87$$

$$\text{GainRatio}(R, B) = 0.002$$

- R: random sample of tuples processed by operator O

$$\text{Entropy}(R) = - \sum_{i=1}^c p_i \ln(p_i)$$

$$\text{InfoGain}(R, A) = \text{Entropy}(R) - \sum_{i=1}^d \frac{|R_i|}{|R|} \text{Entropy}(R_i)$$

$$\text{SplitInformation}(A) = - \sum_{i=1}^d \frac{|R_i|}{|R|} * \log_2 \frac{|R_i|}{|R|}$$

$$\text{GainRatio}(R, A) = \frac{\text{InfoGain}(R, A)}{\text{SplitInformation}(A)}$$

Classifier Attributes: Definition

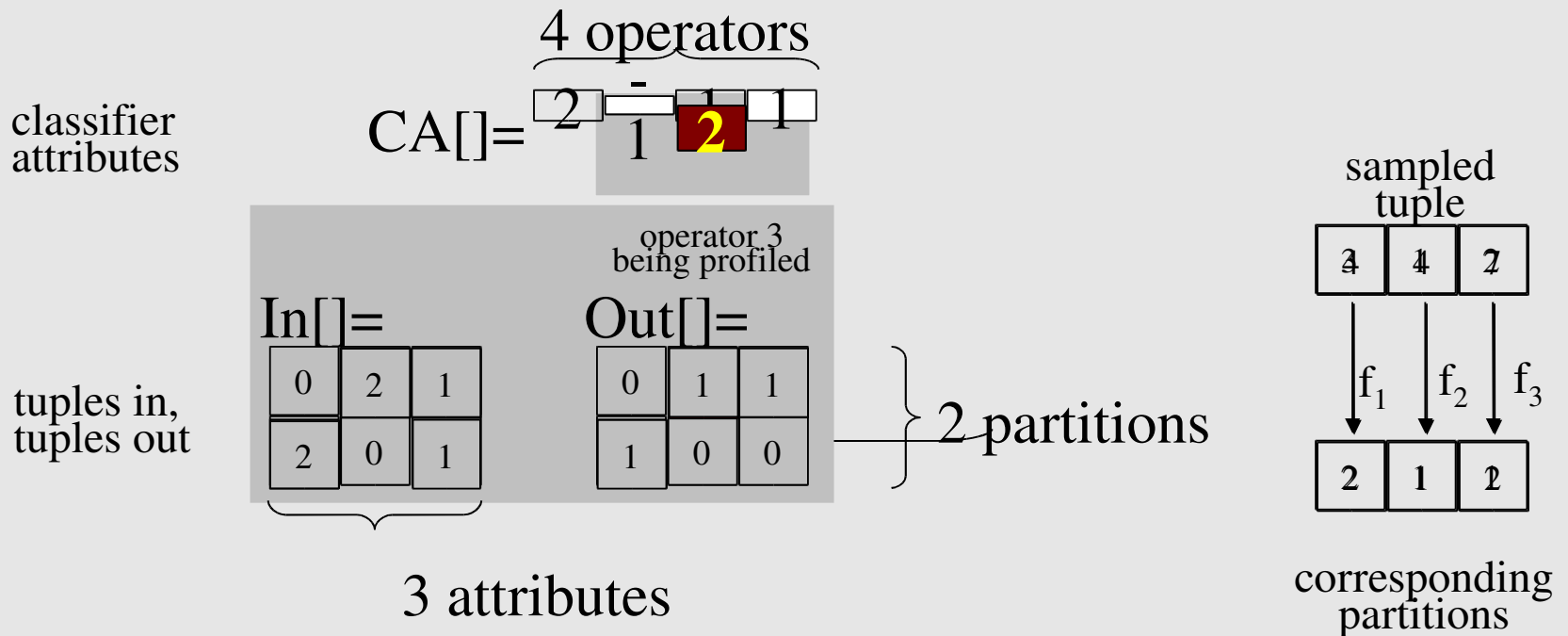
An attribute A is a classifier attribute for operator O , if for any large random sample R of tuples processed by O , $\text{GainRatio}(R, A) > \tau$, for some threshold τ

Content-Learns Algorithm: Learning Routes Automatically

- Content-Learns consists of two continuous, concurrent steps:
 - **Optimization**: For each $O_i \in O_1, \dots, O_n$ find:
 - that O_i does not have a classifier attribute or
 - find the best classifier attribute, C_i , of O_i .
 - **Routing**: Route tuples according to the:
 - selectivities of O_i if O_i does not have a classifier attribute or
 - according to the content-specific selectivities of the pair $\langle O_i, C_i \rangle$ if C_i is the best classifier attribute of O_i

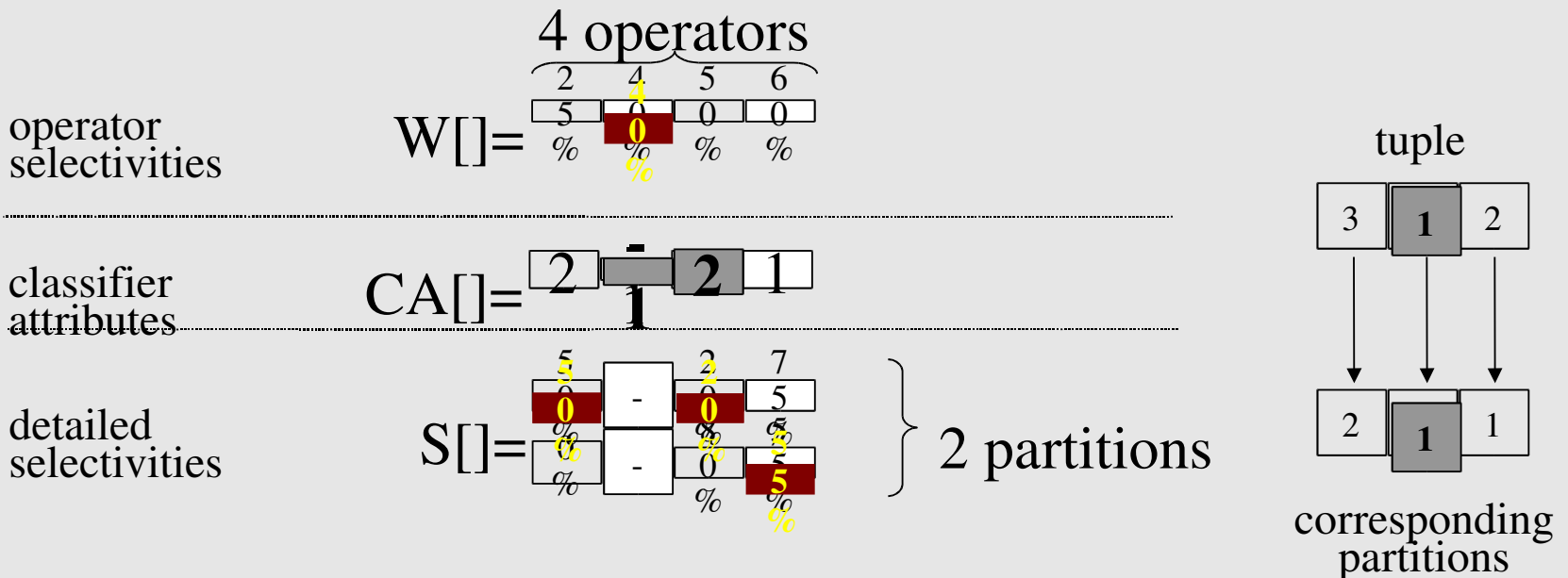
Content-Learns: Optimization Step

- Find C_1 by profiling O_1 :
 - Route a fraction of input tuples to O_1
 - For each sampled tuple
 - For each attribute
 - map attribute values to d partitions
 - update pass/fail counters
 - When all sample tuples seen, compute C_1



Content-Learns: Routing Step

- SBR routes to O_i with probability inversely proportional to O_i 's selectivity, $W[\mathbf{I}]$
- CBR routes to operator with minimum σ :
 - If O_i does not have a classifier attribute, its $\sigma=W[\mathbf{I}]$
 - If O_i has a classifier attribute, its $\sigma=S[\mathbf{I},\mathbf{i}]$, $\mathbf{j}=CA[\mathbf{I}]$, $\mathbf{i}=f_j(t.C_j)$



Adaptivity and Overhead

- CBR introduces new routing and learning overheads
 - Overheads at odds with adaptivity
- Adaptivity: ability to find efficient plan quickly when data or system characteristics change

CBR Update Overheads

- Once per tuple:
 - selectivities as fresh as possible
- Once per sampled tuple:
 - correlations between operators and content
- Once per sample (~2500 tuples)
 - Computing GainRatio and updating one entry in array CA

operator selectivities

	2	4	5	6
W[] =	5	0	0	0
	%	%	%	%

classifier attributes

CA[] =	2	-	2	1
		1		

detailed selectivities

	5	-	2	7
S[] =	0	-	0	5
	%		%	%
	0	-	0	5
	%		%	%

	In[] =	Out[] =	} partitions: <i>1, ..., d</i>
tuples in,	0	0	
tuples out	1	1	
	2	0	
		1	0
		0	0

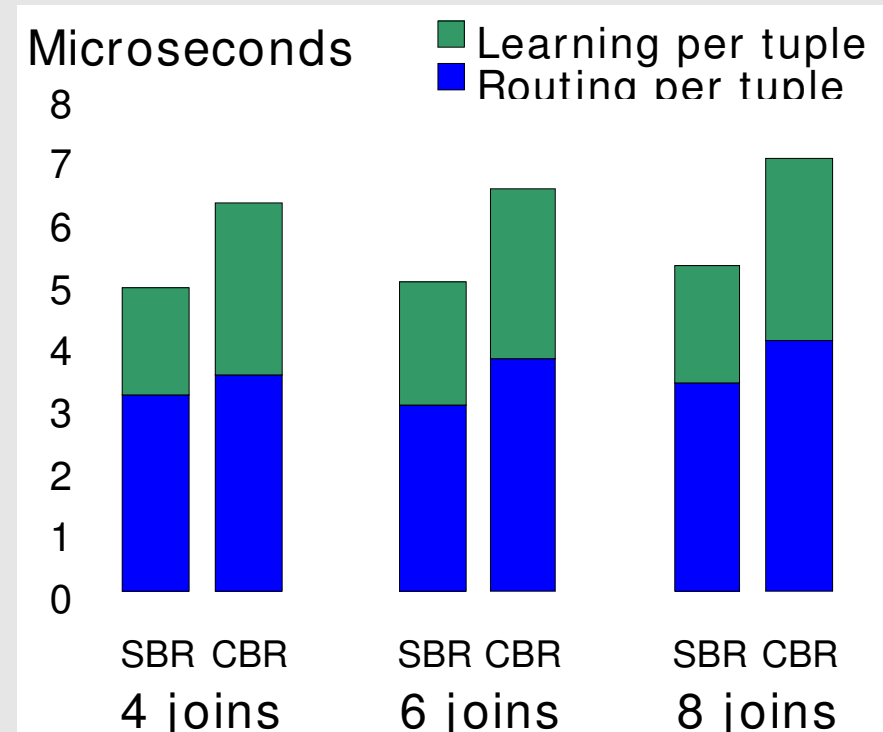
operators: $1, \dots, n$

attributes: $1, \dots, k$

Experimental Results:

Run-time Overheads

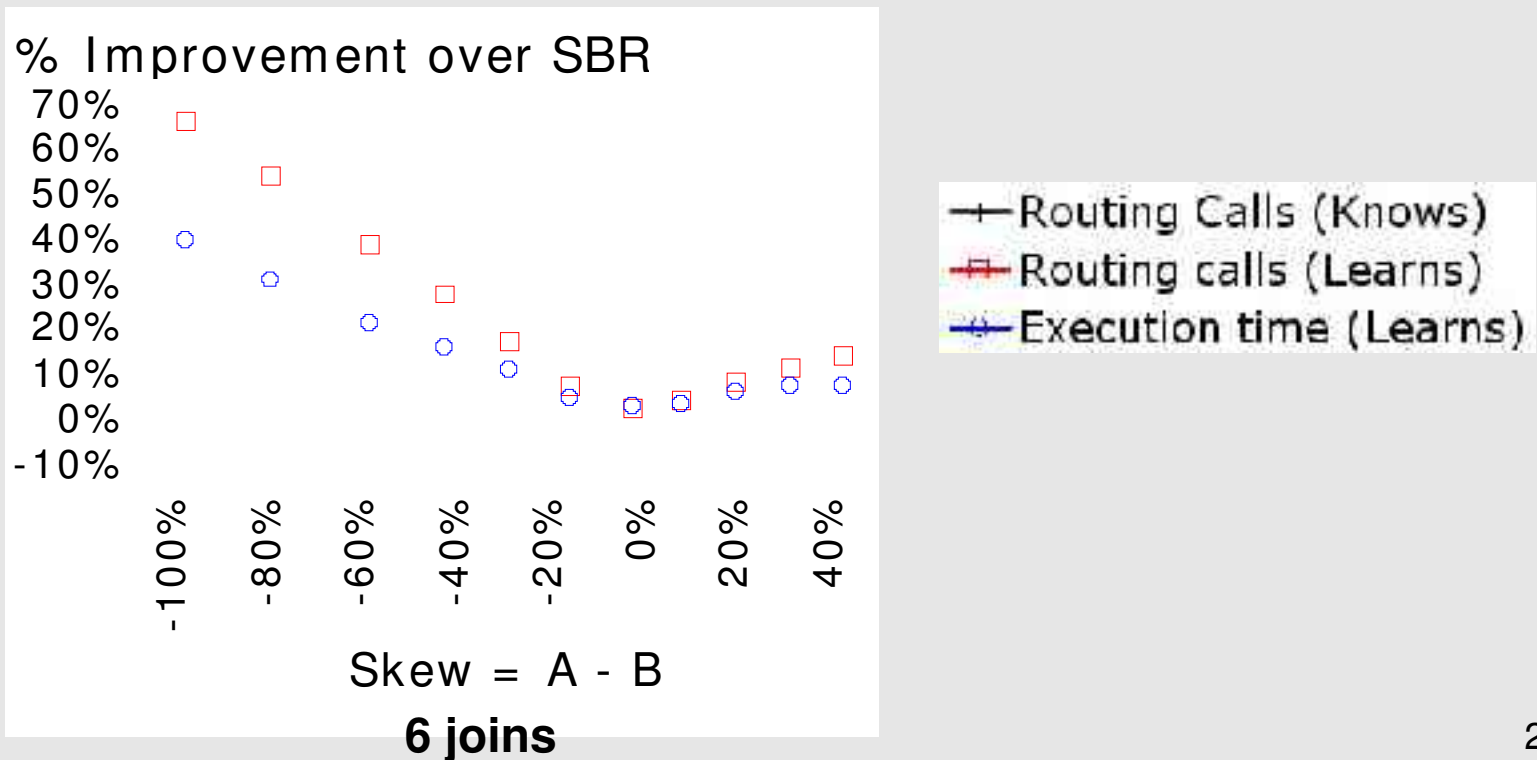
- Routing overhead
 - time to perform routing decisions (SBR, CBR)
- Learning overhead:
 - Time to update data structures (SBR, CBR) plus
 - Time to compute gain ratio (CBR only).



Overhead increase: 30%-45%

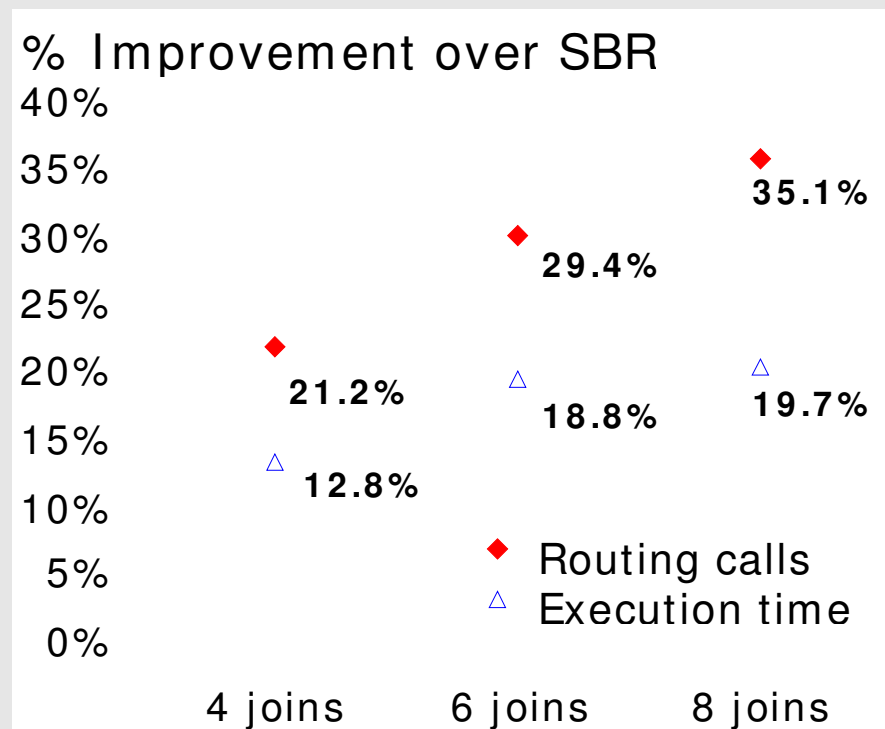
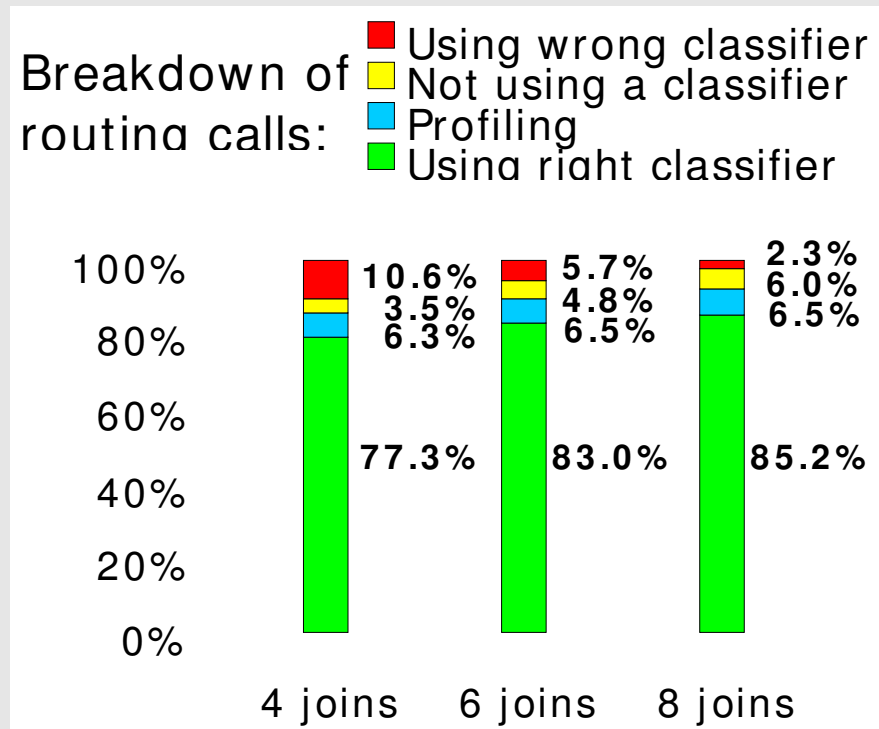
Experimental Results: Varying Skew

- One operator with selectivity A, all others with selectivity B
- Skew is A-B. A varied from 5% to 95%
- Overall selectivity: 5%



Experimental Results: Random Selectivities

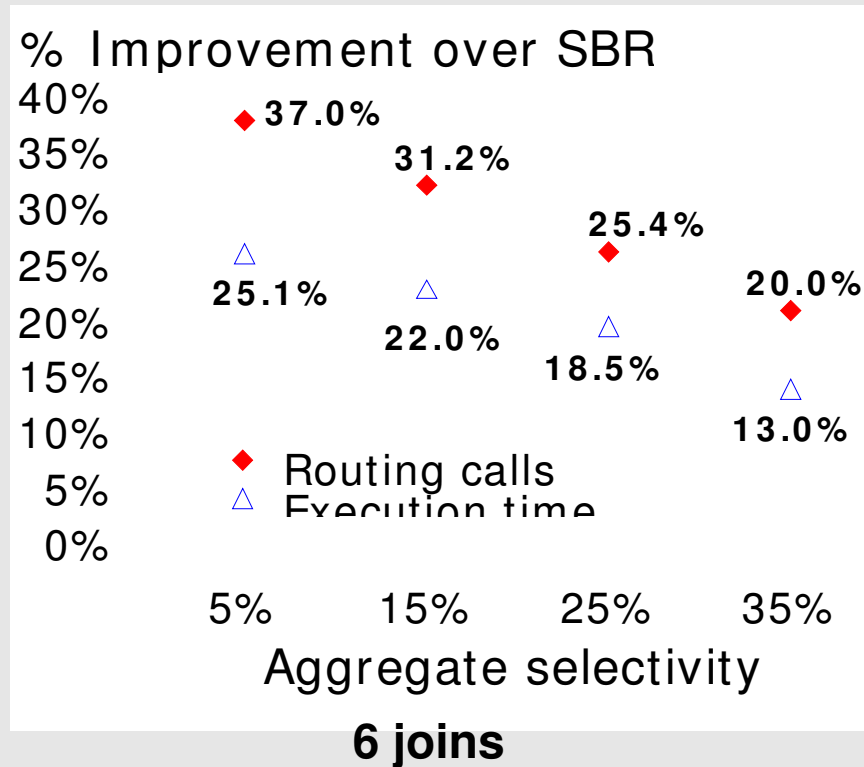
- Attribute *attrC* correlated with the selectivities of the operators
- Other attributes in stream tuples not correlated with selectivities
- Random selectivities in each operator



Experimental Results:

Varying Aggregate Selectivity

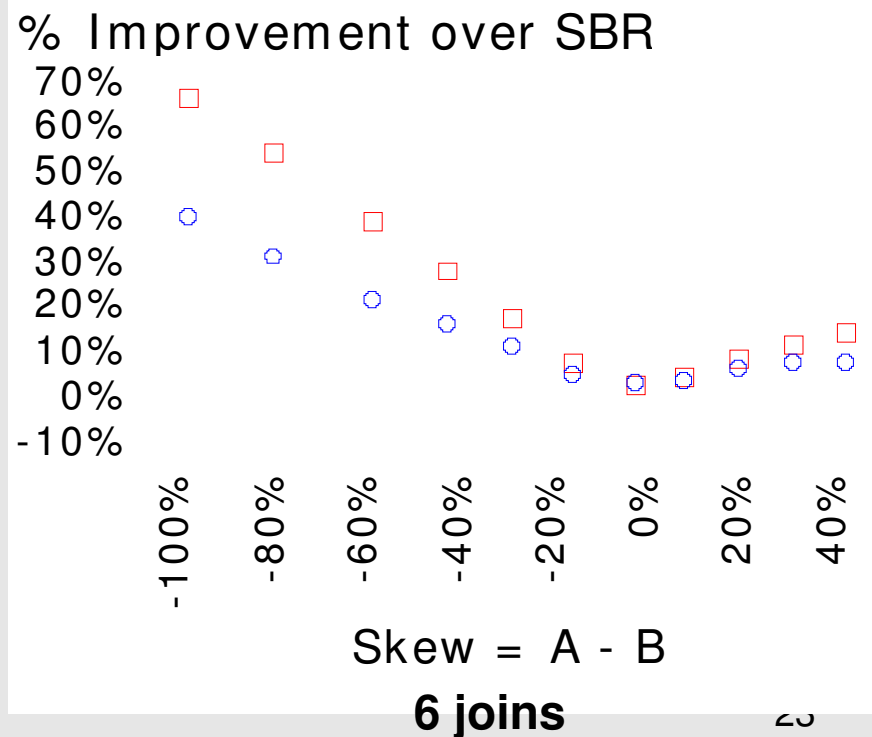
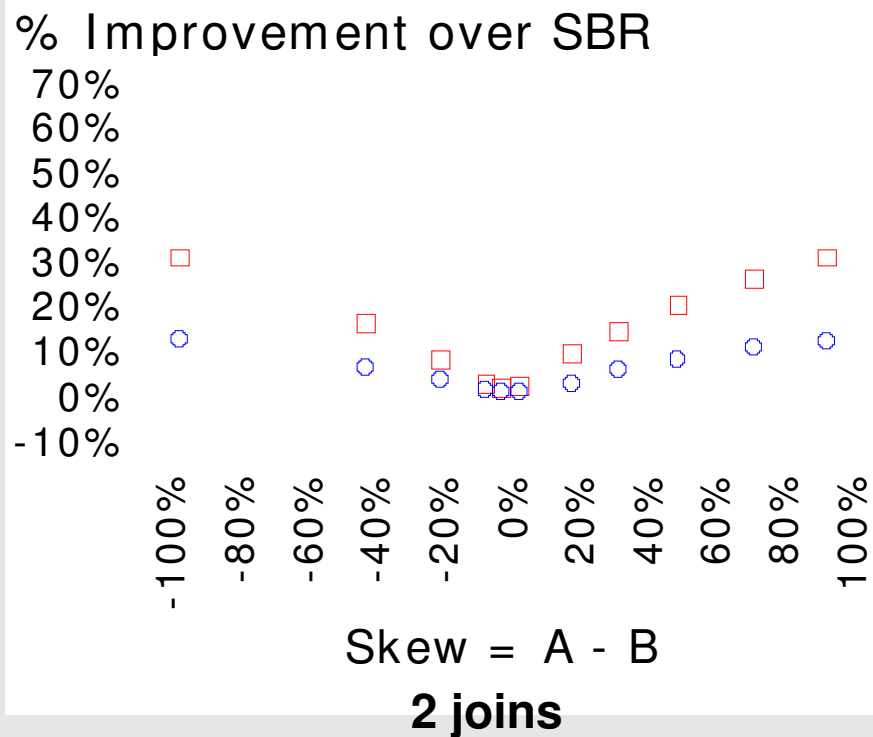
- Aggregate selectivity in previous experiments was 5% or ~8%
- Here we vary aggregate selectivity between 5% to 35%
- Random selectivities within these bounds



Experimental Results: Varying Skew

- One operator with selectivity A, all others with selectivity B
- Skew is A-B. A varied from 5% to 95%
- Overall selectivity: 5%

—+— Routing Calls (Knows)
—+— Routing calls (Learns)
—o— Execution time (Learns)



Conclusions

- CBR eliminates single plan assumption
- Explores correlation between tuple content and operator selectivities
- Adaptive learner of correlations with negligible overhead
- Performance improvements over non-CBR routing