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 WHERE matches(destination, T) AND contains(data, “0xa...8”) 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 \( O_2, O_1, O_3 \)
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 $O_1$, $O_2$, $O_3$

<table>
<thead>
<tr>
<th>Selectivities</th>
<th>$O_1$</th>
<th>$O_2$</th>
<th>$O_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>30%</td>
<td>40%</td>
<td>60%</td>
<td></td>
</tr>
</tbody>
</table>

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 $\{a,b,c\}$

<table>
<thead>
<tr>
<th>Value of $A$</th>
<th>$O_1$</th>
<th>$O_2$</th>
<th>$O_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A=a$</td>
<td>32%</td>
<td>10%</td>
<td>55%</td>
</tr>
<tr>
<td>$A=b$</td>
<td>31%</td>
<td>20%</td>
<td>65%</td>
</tr>
<tr>
<td>$A=c$</td>
<td>27%</td>
<td>90%</td>
<td>60%</td>
</tr>
<tr>
<td>Overall</td>
<td>30%</td>
<td>40%</td>
<td>60%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>A=a</th>
<th>A=b</th>
<th>A=c</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10%</td>
<td>20%</td>
<td>90%</td>
<td>40%</td>
</tr>
<tr>
<td></td>
<td>90%</td>
<td>80%</td>
<td>10%</td>
<td>60%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>B=x</th>
<th>B=y</th>
<th>B=z</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>43%</td>
<td>38%</td>
<td>39%</td>
<td>40%</td>
</tr>
<tr>
<td></td>
<td>57%</td>
<td>62%</td>
<td>61%</td>
<td>60%</td>
</tr>
</tbody>
</table>
**GainRatio to Measure Correlation**

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A=a</td>
<td>10%</td>
<td>90%</td>
<td>B=x</td>
<td>43%</td>
<td>57%</td>
</tr>
<tr>
<td>A=b</td>
<td>20%</td>
<td>80%</td>
<td>B=y</td>
<td>38%</td>
<td>62%</td>
</tr>
<tr>
<td>A=c</td>
<td>90%</td>
<td>10%</td>
<td>B=z</td>
<td>39%</td>
<td>61%</td>
</tr>
<tr>
<td>Overall</td>
<td>40%</td>
<td>60%</td>
<td>Overall</td>
<td>40%</td>
<td>60%</td>
</tr>
</tbody>
</table>

**GainRatio(R, A) = 0.87**  **GainRatio(R, B) = 0.002**

- **R**: random sample of tuples processed by operator O

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

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

\[
SplitInformation(A) = - \sum_{i=1}^{d} \frac{|R_i|}{|R|} \cdot \log_2 \frac{|R_i|}{|R|}
\]

\[
GainRatio(R,A) = \frac{InfoGain(R,A)}{SplitInformation(A)}
\]
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, GainRatio (R, A) > τ, for some threshold τ.
Content-Learns Algorithm: Learning Routes Automatically

• Content-Learns consists of two continuous, concurrent steps:
  – **Optimization**: For each $O_i \in O_1, \ldots, O_n$ 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 $<O_i, C_i>$ if $C_i$ is the best classifier attribute of $O_i$. 
Content-Learns: Optimization Step

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

![Diagram showing tuple routing and attribute partitions]

- Tuples in, Tuples out
- Classifier attributes
- 3 attributes
- 2 partitions
- 4 operators
- Sampled tuple
- Corresponding partitions
Content-Learns: Routing Step

- SBR routes to $O_i$ with probability inversely proportional to $O_i$'s selectivity, $W[1]$
- CBR routes to operator with minimum $\sigma$:
  - If $O_i$ does not have a classifier attribute, its $\sigma=W[1]$
  - If $O_i$ has a classifier attribute, its $\sigma=S[1,i]$, $j=CA[i]$, $i=f_j(t.C_j)$

\[
W[] = \begin{bmatrix}
2 & 4 & 5 & 6 \\
5 & 0 & 0 & 0
\end{bmatrix}
\]

\[
CA[] = \begin{bmatrix}
2 & 1 & 2 & 1
\end{bmatrix}
\]

\[
S[] = \begin{bmatrix}
5 & 0 & 2 & 7 \\
6 & 0 & 5 & 5
\end{bmatrix}
\]

2 partitions
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

\[
W[] = \begin{bmatrix}
2 & 4 & 5 & 6 \\
5 & 0 & 0 & 0 \\
\%
\end{bmatrix}
\]

\[
CA[] = \begin{bmatrix}
2 & - & 2 & 1 \\
\%
\end{bmatrix}
\]

\[
S[] = \begin{bmatrix}
5 & - & 2 & 7 \\
0 & 0 & 5 & 0 \\
\% & \% & \% & \%
\end{bmatrix}
\]

\[
\text{operators: } 1,\ldots,n
\]

\[
\text{attributes: } 1,\ldots,k
\]

\[
\text{tuples in, tuples out: }\]

\[
\text{In[] =}
\begin{bmatrix}
0 & 1 & 2 \\
2 & 1 & 0 \\
\%
\end{bmatrix}
\]

\[
\text{Out[] =}
\begin{bmatrix}
0 & 1 & 1 \\
1 & 0 & 0 \\
\%
\end{bmatrix}
\]

\[
\text{partitions: } 1,\ldots,d
\]
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).

![Graph showing overhead increase: 30%-45%](image_url)
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%

6 joins
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

Breakdown of routing calls:

- Using wrong classifier
- Not using a classifier
- Profiling
- Using right classifier

<table>
<thead>
<tr>
<th></th>
<th>4 joins</th>
<th>6 joins</th>
<th>8 joins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using wrong classifier</td>
<td>10.6%</td>
<td>5.7%</td>
<td>2.3%</td>
</tr>
<tr>
<td>Not using a classifier</td>
<td>3.5%</td>
<td>6.5%</td>
<td>6.0%</td>
</tr>
<tr>
<td>Profiling</td>
<td>77.3%</td>
<td>83.0%</td>
<td>85.2%</td>
</tr>
<tr>
<td>Using right classifier</td>
<td>6.3%</td>
<td>6.5%</td>
<td>6.5%</td>
</tr>
</tbody>
</table>

% Improvement over SBR

- Routing calls: 35.1%
- Execution time: 29.4%, 18.8%, 19.7%
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

![Chart showing % Improvement over SBR vs. Aggregate selectivity for routing calls and execution time for 6 joins]
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%

![Graph showing % Improvement over SBR for 2 joins and 6 joins with varying skew.](image)
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