Rate-Based Query Optimization for Streaming Information Sources

Stratis D. Viglas

Jeffrey F. Naughton

Presented by:

Cansu Aslantas
Streaming Query Optimization

- Cardinality estimation
- Cost estimation
- Adaptive/Dynamic execution
A “General Framework” for (Query) Optimization
A “General Framework” for (Query) Optimization

• Be very careful with
  – Assumptions
  – Over-simplifications
  – Specialization
A “General Framework” for (Query) Optimization

• Be very careful with
  – Assumptions
  – Over-simplifications
  – Specialization

• Wait... Is solving sub-problems bad?
3.2 Selections

For selections we need to incorporate the selectivity of the predicate under evaluation. Given the input rate, the number of input objects in one time unit will be $r_i$. Assuming a uniform distribution, the number of objects appearing in the output will be $f \cdot r_i$, where $f$ is the predicate selectivity. We can calculate the output rate in a way analogous to the calculation of the projection output rate, with the only difference that we are using $C_\sigma$ instead of $C_\pi$.

The output rate will then be $r_o = f \cdot r_i$ if $C_\sigma \leq \frac{1}{r_i}$ and $r_o = \frac{f}{C_\sigma}$ if $C_\sigma > \frac{1}{r_i}$. Again, in most cases it is safe to assume that $C_\sigma \leq \frac{1}{r_i}$, so $r_o = f \cdot r_i.$
Assumption: Selection Cost

3.2 Selections

For selections we need to incorporate the selectivity of the predicate under evaluation. Given the input rate, the number of input objects in one time unit will be \( r_i \). Assuming a uniform distribution, the number of objects appearing in the output will be \( f \cdot r_i \), where \( f \) is the predicate selectivity. We can calculate the output rate in a way analogous to the calculation of the projection output rate, with the only difference that we are using \( C_\sigma \) instead of \( C_\pi \). The output rate will then be \( r_o = f \cdot r_i \) if \( C_\sigma \leq \frac{1}{r_i} \) and \( r_o = f \cdot \frac{r_i}{C_\sigma} \) if \( C_\sigma > \frac{1}{r_i} \). Again, in most cases it is safe to assume that \( C_\sigma \leq \frac{1}{r_i} \), so \( r_o = f \cdot r_i \).

Finally, we note that we made the implicit assumption above that the time to process the tuples arriving during time \( t \), which is \( t \cdot (r_i C_i + r_o C_o) \), is greater than \( t \), which means that \( (r_i C_i + r_o C_o) > 1 \). If this is not the case, the denominator needs to be replaced by 1, since output tuples corresponding to a given input cannot be produced before the input itself arrives. The above holds for Cartesian products as well, with the only modification being that \( f = 1 \).
Over-simplification: Nested Loop Costs

Table 3: Cost formulas for the join algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Left arrival cost ((C_l))</th>
<th>Right arrival cost ((C_r))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested loops</td>
<td>(\text{move} + r_s \cdot t \cdot \text{comp})</td>
<td>(\text{move} + r_R \cdot t \cdot \text{comp})</td>
</tr>
<tr>
<td>Symmetric hash join</td>
<td>(\text{move} + \text{hash} + \text{probe})</td>
<td>(\text{move} + \text{hash} + \text{probe})</td>
</tr>
</tbody>
</table>

- “Non-blocking nested loops has a time-dependent aspect to its cost, so, as time progresses, the cost increases. Symmetric hash join, on the other hand, has a constant cost to handle its inputs.”
Over-simplification: Nested Loop Costs

Table 3: Cost formulas for the join algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Left arrival cost ($C_l$)</th>
<th>Right arrival cost ($C_r$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested loops</td>
<td>$\text{move} + r_s \cdot \text{comp}$</td>
<td>$\text{move} + r_R \cdot \text{comp}$</td>
</tr>
<tr>
<td>Symmetric hash join</td>
<td>$\text{move} + \text{hash} + \text{probe}$</td>
<td>$\text{move} + \text{hash} + \text{probe}$</td>
</tr>
</tbody>
</table>

- “Non-blocking nested loops has a time-dependent aspect to its cost, so, as time progresses, the cost increases. Symmetric hash join, on the other hand, has a constant cost to handle its inputs.”

- Is it only the cost that’s increasing?
Specification: Symmetric Hash Join

• All experimental setup built on symmetric hash joins.

1. Does the cost model correctly estimate individual plan performance?
2. Is the framework capable of providing correct decisions regarding the best choice among a set of plans?

• I don’t know.
Making good estimates

• ... is one thing.
Making good estimates

• ... is one thing.

• How long *it takes* to make those estimates is another.
  – How does it compare to the Related Work?
Conclusion

• Presented as a general framework.
• Only proven to work for symmetric hash joins.
  – Either use more formal proofs,
  – Or experiment all (more) cases.
• Performance unknown.