Robust Query Optimization Methods With Respect to Estimation Errors: A Survey

Shaoyi Yin, Abdelkader Hameurlain, Franck Morvan
Architecture of query processing

Keywords:
- Cardinality estimates
- Cost model
- Optimal plan
- Robustness
So what exactly is the problem with current query engines?

Since it’s not realistic to maintain accurate cardinalities in the database catalog, the optimizer calculates costs of different plans with incorrect values. The system might think one plan is optimal, but in reality, it may be quite the opposite.
Comparison criteria used in the paper

- C1: Estimation Error Sources
- C2: Target Query Types
- C3: Target Optimization Decisions
- C4: Performance Degradation Risk
- C5: Engineering Cost
Main categories for strategies

Single plan:
- Cardinality injection
- Plan modification
- Robust plan selection

Multi plan:
- Deferred Plan Choosing
- Tuple Routing through Eddies
- Optimizer Controlled Data Partitioning
Single plan
Cardinality injection

- Obtains cardinality value directly after an operator has been executed.
- Uses values from previous queries to make optimization decisions.
- Some approaches may execute crucial sub-queries during optimization to acquire accurate cardinalities for the current query.
Plan modification

- Re-optimization
  - Stops execution of current plan and returns to the optimizer to generate a new one
  - The optimizer will use cardinalities obtained from previous execution to generate new plan

- Rescheduling
  - Re-orders operators if a delay is noticed

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Scrambling</td>
<td>AD</td>
<td>C</td>
<td>OEO,JO</td>
<td>H</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>DQS</td>
<td>AD,AR</td>
<td>C</td>
<td>OEO,JO</td>
<td>L</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>ReOpt</td>
<td>DD,DC,DM</td>
<td>C</td>
<td>JM,JO,MA</td>
<td>H</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>POP</td>
<td>DD,DC,DM</td>
<td>C</td>
<td>AM,JM,JO</td>
<td>H</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>CQO</td>
<td>MS</td>
<td>C</td>
<td>PD,PK</td>
<td>L</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>LAW</td>
<td>DD,DC,DM,MS</td>
<td>C</td>
<td>AM,JM,JO</td>
<td>H</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>IMA</td>
<td>DD,DC,DM,UR</td>
<td>C</td>
<td>CA,MA,PD</td>
<td>H</td>
<td>L</td>
<td></td>
</tr>
</tbody>
</table>
Robust plan selection

- Creates one static, robust plan
  - The plan is never modified after execution starts
- Tries to find a plan that will never give worst-case performance
- Has a predictability/performance trade-off
Multi plan
Deferred Plan Choosing

- Defers choice of plan until absolutely necessary
- Optimized plan may be partly finished when execution starts

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Method</th>
<th>C1: Estim. error sources</th>
<th>C2: Target</th>
<th>C3: Target etc. opt. decisions</th>
<th>C3: Target</th>
<th>C4: Decision risk</th>
<th>C5: Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMC</td>
<td>DD, DC, DM, MS</td>
<td>C</td>
<td>AM</td>
<td>LC</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP</td>
<td>DD, DC, DM, MS</td>
<td>C</td>
<td>JM, JO</td>
<td>LC</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CP</td>
<td>DD, DC, DM, MS</td>
<td>P</td>
<td>AM, JM, JO</td>
<td>L</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PB</td>
<td>DD, DC, DM, MS</td>
<td>P</td>
<td>AM, JM, JO</td>
<td>L</td>
<td>M</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Number of rows $r$: 287

DEFERRED PLAN

Choosing

- Defers choice of plan until absolutely necessary.
- Optimized plan may be partly finished when execution starts.
Tuple Routing through Eddies

- Data from tables are divided into tuples
- The Eddy routes tuples to the different operators
- Each tuple keeps a bitmap that tracks which operator it has visited and has yet to visit
- The plan is pre-optimized before the Eddy does the final optimization
Optimizer Controlled Data Partitioning

- Data is divided into partitions based on their characteristics
- Each partition can execute a different plan
- Examples:
  - Change plan for data if the first plan does not perform as expected
  - Partly sorted data can use one plan for the sorted part and another one for the other
  - Plans can be executed in parallel to find the fastest one
## Global comparison

<table>
<thead>
<tr>
<th>Approach</th>
<th>Strategy</th>
<th>Advantage</th>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Plan Based</td>
<td>CI</td>
<td>Good for repeatedly-running queries</td>
<td>For current query, only JM, JO are optimized</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>Could be extended to improve all kinds of opt. decisions</td>
<td>May have high degradation risk</td>
</tr>
<tr>
<td></td>
<td>RPS</td>
<td>Degradation risk is low if a robust plan exists</td>
<td>Difficult to handle too many uncertain factors</td>
</tr>
<tr>
<td>Multi-Plan Based</td>
<td>DPC</td>
<td>Easier to implement than TR and DP</td>
<td>AMC may consume too many resources</td>
</tr>
<tr>
<td></td>
<td>TRE</td>
<td>Deal with all kinds of estimation error sources</td>
<td>Memory consuming; Parallelization problem not addressed</td>
</tr>
<tr>
<td></td>
<td>OCDP</td>
<td>Take advantage of inherent data characteristics</td>
<td>Optimization time may be long</td>
</tr>
</tbody>
</table>
Questions?