SAMPLING-BASED QUERY RE-OPTIMIZATION

Wentao Wu
Microsoft Research
Query optimization remains challenging despite of decades of efforts and progresses.

Cardinality estimation is the key challenge.
- Selectivity of join predicates
- Correlation of columns
Histogram vs. Sampling

- Single-column histograms cannot capture data correlations between columns.
  - Use the attribute-value-independence (AVI) assumption.

- Sampling is better than histograms on capturing data correlations.
  - We run query over exact rather than summarized data.
But Why are Histograms Dominant?

- The overhead is much smaller, compared with other cardinality estimation approaches.

- Sampling incurs additional overhead and should be used conservatively.
  
  A naïve idea: use sampling for all plans considered by the optimizer.
Cost-Based Query Optimization

Pick the best plan from $N$ candidates:

For large $N$, sampling is *not* affordable to be used for every plan.

$N$ could be large! ($10^2$ or even $10^3$)
Our Idea

- Use sampling as a post-processing validation step.
  - Detect cardinality estimation errors for the final plan returned by the optimizer.

- Re-optimize the query if cardinality estimation errors are detected.

Catch big mistakes of the optimizer before the plan runs!
The Re-optimization Algorithm

Query $q$ → Query Optimizer → Plan $P_q$ → Final Plan

- Update Cardinalities
- Refined Cardinality Estimates $\Gamma$
- Sampling-based Validation
The Re-optimization Algorithm (Cont.)

Example: $q = A \bowtie B \bowtie C$

Query $q$

Query Optimizer

Join | Cardinality
---|---
$A \bowtie B$ | 100
$B \bowtie C$ | 300
$A \bowtie C$ | 500

Update

$A \bowtie B: 1000$

Sampling-based Validation

(Final Plan)
Efficiency of Re-optimization

- The worst-case expected number of iterations:
  \[ S_N = \sum_{k=1}^{N} k \cdot \left(1 - \frac{1}{N}\right) \cdots \left(1 - \frac{k-1}{N}\right) \cdot \frac{k}{N} \]

- \( S_N \sim O(\sqrt{N}) \).

\( N \) is the number of *join trees* in the search space.
Quality of Re-optimized Plans

- If sampling-based cost estimates are consistent with the actual costs, that is,

  \[ \text{cost}_{\text{est}}(P_1) < \text{cost}_{\text{est}}(P_2) \implies \text{cost}_{\text{act}}(P_1) < \text{cost}_{\text{act}}(P_2), \]

  then the final re-optimized plan is \textit{locally optimal}:

  \[ \text{cost}_{\text{act}}(P_{\text{final}}) \leq \text{cost}_{\text{act}}(P), \text{ for any } P \text{ in re-optimization.} \]

- However, cost models are imperfect, and cardinality estimates based on sampling are imperfect, too.
  - See experimental results.
Experimental Evaluation

- We implemented the re-optimization procedure in PostgreSQL 9.0.4.

- We have two goals:
  - Test the approach for “common” cases.
  - Test the approach for “corner” cases.
Experimental Evaluation (Cont.)

- “Common” cases
  - 10GB TPC-H benchmark

- “Corner” cases
  - (Homegrown) Optimizer “Torture Test” (OTT)

Specially designed database and queries with high data correlation that can challenge query optimizers.
Experimental Evaluation (Cont.)

- Results on the 10GB TPC-H database
Results of the “torture test” (5-join queries, log-scale)

- Original Plan
- Re-optimized Plan

Running Time (s)

OTT Query
More details about OTT:

- K tables $R_1, \ldots, R_K$, with $R_k(A_k, B_k)$
- Each $R_k$ is generated independently, with $B_k = A_k$.
- $A_k$ (and thus $B_k$) is uniformly distributed.
- The queries look like:

$$\sigma_{A_1=c_1 \land \cdots \land A_K=c_K \land B_1=B_2 \land \cdots \land B_{K-1}=B_K}(R_1 \times \cdots \times R_K)$$

**Property**: These queries are not empty if and only if $A_1 = \ldots = A_K$!
An instance of OTT used in our experiments:

- Use 6 TPC-H tables (excluding “nation” and “region”).
- Use a set of _empty_ queries with _non-empty_ sub-queries.

**Non-empty**

```
\( s.B = o.B \)
\( o.B = c.B \)
\( s.A = 0 \)
\( o.A = 0 \)
```

**customer**
```
\( c.A = 1 \)
```

**supplier**
```
\( s.A = 0 \)
```

**order**
```
```

**Empty!**

```
\( c.B = o.B \)
\( o.B = s.B \)
```

**customer**
```
\( c.A = 1 \)
```

**supplier**
```
\( s.A = 0 \)
```

**order**
```
```

**Bad Plan**

**Good Plan**
Summary

Sampling as post-processing: efficiency/effectiveness tradeoff!
Q & A

- Thank you😊
Cardinality Estimation Methods

- **Histograms**
  - Single-column histograms (dominant in current DBMS)
  - Multi-column histograms

- **Other methods**
  - Offline approaches: sampling, sketch, graphical models
  - Online approaches: dynamic query plans, parametric query optimization, query feedback, mid-query re-optimization, plan bouquets
Estimate the selectivity $\rho_q$ of a join query $q = R_1 \bowtie R_2$.


Do a “cross product” over the samples: $\rho(i,j) = 0$ or $1$.

The estimator $\hat{\rho}_q$ is unbiased and strongly consistent.
Other Sampling-Based Methods

- Sampling-Based Estimation of the Number of Distinct Values of an Attribute, VLDB’95
- Towards Estimation Error Guarantees for Distinct Values, PODS’00
- End-biased Samples for Join Cardinality Estimation, ICDE’06
- Join Size Estimation Subject to Filter Conditions, VLDB’15
Convergence of Re-optimization

- Convergence Condition of Re-optimization

**Theorem:** The re-optimization procedure terminates when all the joins in the returned query plan have been observed in previous rounds of iteration.

For example, re-optimization will terminate after $T_1'$ is returned.
The previous convergence condition is *sufficient* but *not necessary*. Re-optimization could terminate even before it meets the previous condition.

To understand re-optimization better, we need the notion of local/global transformations.
Local/Global Transformations

- Local transformation of query plans

Local transformations are those plans that share the same joins. They only differ in choices of specific physical operators.
Characterization of Re-optimization

- The three possible cases in re-optimization:
  - (1) It terminates in two steps with $P2 = P1$.
  - (2) It terminates in $n + 1$ steps ($n > 1$) where all plan transitions are *global* transformations.
  - (3) It terminates in $n + 1$ steps ($n > 1$) where only the *last* transition is a *local* transformation: the others are all global transformations.
An illustration of Case (2) and (3):

Case (2): $P_1 \xrightarrow{g} \cdots \xrightarrow{g} P_{n-1} \xrightarrow{g} P_n = P_{n+1}$

Case (3): $P_1 \xrightarrow{g} \cdots \xrightarrow{g} P_{n-1} \xrightarrow{l} P_n = P_{n+1}$

The number of iterations thus depends on the number of global transformations!
A probabilistic model for analysis of expected number of steps in re-optimization:

- We have $N$ balls in a queue, initially unmarked.

The probability that the ball will be inserted at any position in the queue is uniformly $1/N$. 
The expected number of steps of the previous procedure is:

\[ S_N = \sum_{k=1}^{N} k \cdot \left(1 - \frac{1}{N}\right) \cdots \left(1 - \frac{k-1}{N}\right) \cdot \frac{k}{N} \]

How is it related to query optimizations?
- Think of query plans (or, globally different join trees) as balls!

The uniform distribution employed in the model may be invalid in practice.
- We have more analysis for situations where underestimation or overestimation is dominant. (And more analysis could be done in the future.)