Probabilistic Robust Query Optimization

Rohan Nogueira

March 31, 2007

Outline

Motivation

Cardinality Estimation

Making Uncertainty Explicit

Analysis

Experiments

Related Work [Chu et al, 2002]

Conclusion

Motivation

- Current focus of optimizers is on the best plan possible
- However, consistency and predictability is ignored
- Also, cardinality estimation errors might exist

Current issues

- Attribute Value Independence(AVI) assumption rarely holds
- Problems with many dimensions("Curse of Dimensionality")

Performance v/s Predictability

- Some plans might always be slow (e.g. Sequential Scans)
- Others, might be fast for some range of selectivities, slow otherwise (e.g. Algorithms related to indexes)
- Ideally, the user should be free to decide what he wants

Solving the uncertainty problem

- Rather than giving point estimates, probaility distributions would be a better idea
- Then, select plan with least expected cost, not just least cost
- Can just call the optimizer several times for different selectivities, a waste of time

Incorporating the Probability Distribution

- How do we decide between 2 plans given their probability distributions?
- User decides on performance v/s predictability
- User specifies confidence threshold X%
- The plan is picked which has lower cost X% of the time

Selectivity Estimation via Sampling

- Use precomputed random samples to guess the probability distribution for the selectivity
- For joins create "Join Synopses" (limited to joins on foreign keys)
- Avoids build-up of estimation errors due to AVI

Join Synopses

- The join synopsis for a relation R is constructed as follows:
    1. Construct a uniform random sample from R
    2. For every relation S such that R has a foreign key to S, join the sample of R with S
    3. Repeat step 2 recursively
- This is performed while updating statistics

Deriving the Probability Distribution

- We use Bayes' rule to estimate the probability of a predicate succeeding given the observations on the sample set, X, a vector $\langle x_1, x_2, \ldots, x_n \rangle$
- The equation becomes:

$$f(z|X) = \frac{P[X|p = z]f(z)}{\int_0^1 P[X|p = y]f(y)dy} \tag{1}$$

- $f(z)$ is not always given. We can estimate it given the workload, or use the uniform prior i.e. $f(z) = 1$ or Jeffreys' prior i.e. $f(z) \propto z^{-1/2}(1 - z)^{-1/2}$

Deriving the Probability Distribution(contd.)

- Suppose k tuples satisfy the predicate
- The fraction of tuples satisfying the predicate is p, and that of those that don't is $1 - p$
- The variables x_i are independent and identically distributed Bernoulli random variables, so $Pr[X|p = z] = z^k(1 - z)^{n-k}$
- Combining the above equation with $f(z) = 1$, we get

$$f(z|X) = \frac{z^k(1 - z)^{n-k}}{\int_0^1 y^k(1 - y)^{n-k} dy} \tag{2}$$

- The denominator is independent of z, and may be treated as a normalizing constant
- Thus, it is the beta distribution with parameters $(k + 1, n - k + 1)$

The estimation procedure

- The estimation procedure can be summed up as follows:
  1. Select the correct sample based on relations present in the query
  2. Evaluate the predicate on the sample and use Bayes' rule to infer probability distribution
  3. Choose confidence threshold, T based on user preference, and assign $s = cdf^{-1}(T)$
  4. Return s as the selectivity for the predicate

The estimation procedure (contd.)

- T will be smaller if the user prefers a more aggressive approach, larger for a more predictive approach
- Sample size is most important factor in determining the distribution.

Building the Analytical model

- Consider a query Q, with 2 possible plans P_1 and P_2, running on a table with N rows
- Either plan may be optimal given the selectivity of the query
- We assume a linear cost model of the form $v_i x + f_i$ for plan P_i, with v_i being the cost per tuple and f_i, the fixed overhead per execution

Analytical results

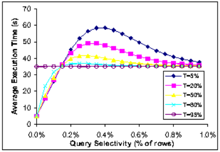

Figure: Confidence Threshold effect

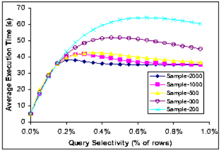

Figure: Effect of Sample Size

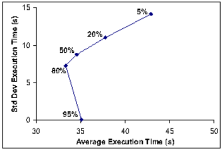

Figure: Performance v/s Predictability

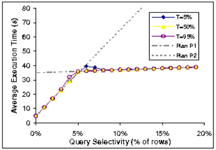

Figure: Crossover point at high selectivity

Experimental results

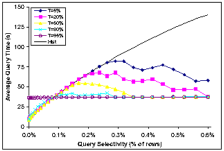

Figure: Selectivity v/s Time

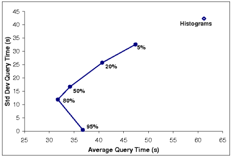

Figure: Performance v/s Predictability

Figure: Results for a query with 2 predicates

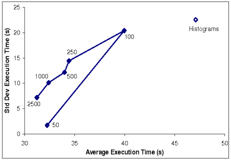

Figure: Effect of varying sample size

Related Work [Chu et al, 2002]

- [Chu et al, 2002] also looks at using Least Expected Cost(LEC)
- It is shown that LEC can be applied not just to cost functions related to running time, but also to more general cost functions
- In addition, the conditions under which current optimizers can produce LEC plans are investigated

When do we get LEC plans

- We can obtain LEC plans if there exists a parameter setting that gives an LEC plan
- Some conditions help us find such a parameter efficiently, such as
  - The presence of a dominant plan
  - The cost of a plan being linear in the parameters of interest
  - The cost of a plan being the sum of products of independent parameters
- Parameters that don't fit these criteria can be transformed so that they do

More general cost functions

- Cost functions that aren't necessarily a function of running time can be easily evaluated using LEC
- However, current optimizers can't be used

Least expected user cost query optimization

- Dynamic programming algorithms like System R are considered
- If the cost function is additive, an LEC plan is produced
- Can modify System R to produce LEC plan
- Can also include variance in the cost function

Conclusion

- Estimation of cardinality distribution using random sampling of relation
- User-defined threshold decides cardinality estimate for optimization
- However, evaluated only for limited conditions
- Might be possible to use LEC optimization with current techniques, but only in some cases

References

Babcock, B. and Chaudhuri, S. (2005) Towards a Robust Query Optimizer: A Principled and Practical Approach ACM SIGMOD, 2005

Chu, F., Halpern, J. and Gehrke, J. (2002) Least Expected Cost Query Optimization: What can we expect? ACM PODS, 2002