Statistical Learning Techniques for Costing XML Queries

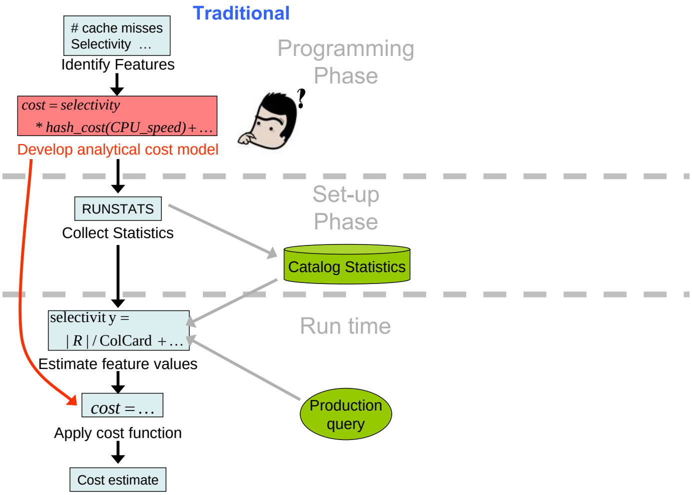
Ning Zhang[1] Peter J. Haas[2] Vanja Josifovski[2]
Guy M. Lohman[2] Chun Zhang[2]

[1]University of Waterloo

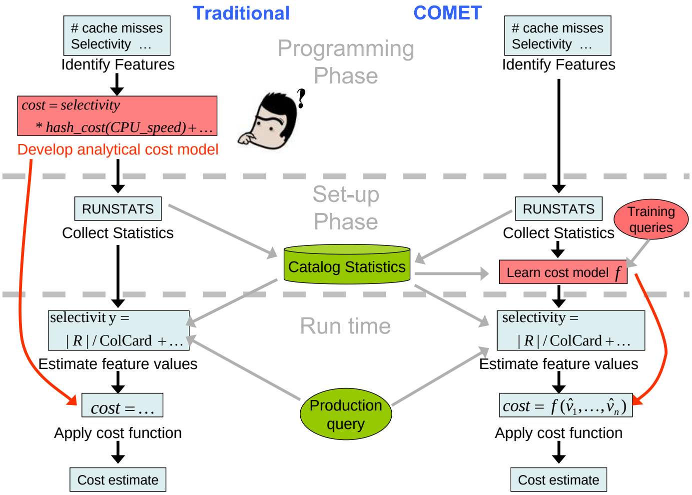
[2]IBM Almaden Research Center

VLDB 2005

COMET: A New Cost-Modeling Approach



COMET: A New Cost-Modeling Approach



Advantages of COMET Approach

Can handle complex operators using statistical learning

- Operators not decomposable into simple scans, joins, etc.
- Operators with highly non-sequential data access patterns
- Used successfully to cost UDFs, remote DB systems (Lee et al. 2004, He et al. 2004, Rahal et al. 2004)

Advantages of COMET Approach

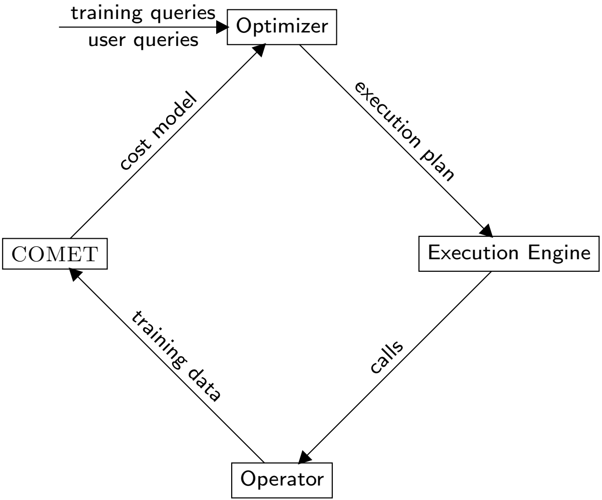
Can handle complex operators using statistical learning

- Operators not decomposable into simple scans, joins, etc.
- Operators with highly non-sequential data access patterns
- Used successfully to cost UDFs, remote DB systems (Lee et al. 2004, He et al. 2004, Rahal et al. 2004)

Simplifies cost-model development

- Reduces need for painstaking code analysis used in analytical modeling
- Easier to incorporate new operators into optimizer
- Helps avoid brittle simplifying assumptions
- Avoids need to explicitly incorporate HW parameters

COMET Permits Optimizer to be Self Tuning



Our Motivation: XML Query Optimization

Query q_1:

```
<bib>
{
  for $b in
    doc("bib.xml")/bib/book
  where
   $b/authors//last = "Stevens"
   and $b/@year > 1991
  return
    <book>
      { $b/title }
    </book>
}
</bib>
```

Need to cost candidate execution plans:

1. Navigational plan:
   - navigate the `bib.xml` tree
   - check pred's for each **book**
2. Value-based index plan:
   - find elements with "Stevens" or "1991" using value-based index
   - navigate up to **book** and check remaining conditions
3. Structure-based index plan:
   - look up matching tree structures using a path/twig index
   - check pred's for each **book**

Today's Talk: Application of COMET Approach to an XML Operator

XML operator to be modeled:

- XNAV operator (complex and dynamic, so hard to model)
- Adaptation of TurboXPath (Josifovski et al. 2005)
- Will model CPU costs (nontrivial component of overall cost)
  - prior work has focused primarily on cardinality estimation

Today's Talk: Application of COMET Approach to an XML Operator

XML operator to be modeled:

- XNAV operator (complex and dynamic, so hard to model)
- Adaptation of TurboXPath (Josifovski et al. 2005)
- Will model CPU costs (nontrivial component of overall cost)
  - prior work has focused primarily on cardinality estimation

Nontrivial steps in applying COMET methodology:

Step 1: Identify XNAV features

Step 2: Determine statistics for estimating feature values

Step 3: Determine formulas for feature-value estimation

Step 4: Identify appropriate statistical learning algorithm for fitting cost model

XNAV: A Complex XML Navigational Operator

What is XNAV?

- $XNAV_{XPath}(XMLTrees) \longrightarrow$ list of matching XML nodes
- XNAV is complex:
  - equivalent to non-decomposable N-way join
  - data stored as paged tree

High-level description of XNAV algorithm:

- XNAV traverses the XML tree in a single pass, with possible skipping of nodes
- XNAV maintains internal states and buffers for matching the query tree during the traversal

Step 1: Identifying XNAV Features

Basis for feature identification

- Knowledge of XNAV algorithm (involves human interaction)
- Trial and error experimentation (with cross-validation)

Step 1: Identifying XNAV Features

Basis for feature identification

- Knowledge of XNAV algorithm (involves human interaction)
- Trial and error experimentation (with cross-validation)

Learning algorithm automatically removes redundant features

- Just need to find "at least enough" features

Step 1: Identifying XNAV Features

Basis for feature identification

- Knowledge of XNAV algorithm (involves human interaction)
- Trial and error experimentation (with cross-validation)

Learning algorithm automatically removes redundant features

- Just need to find "at least enough" features

Some features for XNAV:

- **#visits** : # of XML nodes actually traversed
- **#p_requests** : # of pages read
- ... more features given in the paper

Step 2: Novel Statistics for Estimating Features

How to choose statistics ?

- "As simple as possible, but not simpler"
  - Easy to collect and maintain, less error-prone
- Need to balance space and time requirements
  - Storing redundant stats can speed up feature-value estimation

Step 2: Novel Statistics for Estimating Features

How to choose statistics ?

- "As simple as possible, but not simpler"
  - Easy to collect and maintain, less error-prone
- Need to balance space and time requirements
  - Storing redundant stats can speed up feature-value estimation

Example — Simple Path (SP) Statistics

- **cardinality**: $|p|$, where p is a "simple" path (no branching, no wildcards, etc.)
- **children and descendant cardinality**: $|p/*|$ and $|p//*|$
- **page cardinality**: $\|p\|$
- ... more in the paper

Step 3: Feature-Value Estimation Using Stats

Can estimate all needed feature values using SP stats

- Analysis required, but much easier than analyzing entire XNAV operator
- See paper for detailed formulas (algorithms)
- Formulas tend to overestimate feature values, but COMET automatically compensates for bias (see below)

Step 3: Feature-Value Estimation Using Stats

Can estimate all needed feature values using SP stats

- Analysis required, but much easier than analyzing entire XNAV operator
- See paper for detailed formulas (algorithms)
- Formulas tend to overestimate feature values, but COMET automatically compensates for bias (see below)

Example

- **#visits** $= \sum_{p \in S} |p/*| + \sum_{q \in C} |q//*|$

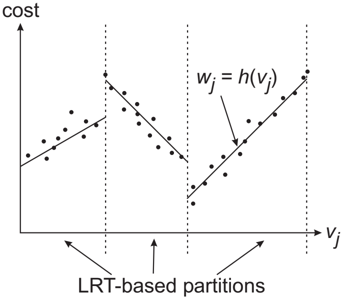
  where S is a set of root-to-non-leaf simple path in the query tree whose next step is connected by a /-axis;
  C is a set of root-to-non-leaf simple path in the query tree whose next step is connected by a //-axis

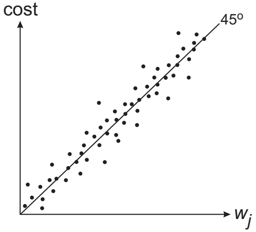
Step 4: Fitting The Cost Model

Use Transform Regression (Pednault 2004)

- "Linear regression on steroids"
- Handles discontinuities and nonlinearities in cost function
- Fully automated (no statistician needed) and highly efficient
- Seamlessly handles both numerical and categorical features

Uses 1-level linear regression tree to "linearize" each feature





Step 4: Transform Regression—Continued

Uses multivariate linear regression on linearized features

- Greedy forward stepwise-regression
- Handles redundant features (multicollinearity)

Step 4: Transform Regression—Continued

Uses multivariate linear regression on linearized features

- Greedy forward stepwise-regression
- Handles redundant features (multicollinearity)

Uses "gradient boosting" to capture feature interactions

- First-order model: models the cost
- ith-order model: models the error in $(i-1)$st-order model

Step 4: Transform Regression—Continued

Uses multivariate linear regression on linearized features

- Greedy forward stepwise-regression
- Handles redundant features (multicollinearity)

Uses "gradient boosting" to capture feature interactions

- First-order model: models the cost
- ith-order model: models the error in $(i-1)$st-order model

Uses other tricks to speed up convergence and improve the fit

- See paper for details

Step 4: Transform Regression—Continued

Uses multivariate linear regression on linearized features

- Greedy forward stepwise-regression
- Handles redundant features (multicollinearity)

Uses "gradient boosting" to capture feature interactions

- First-order model: models the cost
- ith-order model: models the error in $(i-1)$st-order model

Uses other tricks to speed up convergence and improve the fit

- See paper for details

Model learned from estimated feature values

- So COMET is robust to systematic bias in feature-value estimation

Experimental Study

Training data and queries:

- Synthetic and real-world data sets
  (Including TPC-H, XMark, NASA, and XBench)
- Randomly generated queries:
  - Simple linear paths (e.g., `/a/b/c`)
  - Branching paths (e.g., `/a[b][c]/d`)
  - Complex paths (e.g., `/a[.//b][c//d]//e`)

Experimental Study

Training data and queries:

- Synthetic and real-world data sets
  (Including TPC-H, XMark, NASA, and XBench)
- Randomly generated queries:
  - Simple linear paths (e.g., `/a/b/c`)
  - Branching paths (e.g., `/a[b][c]/d`)
  - Complex paths (e.g., `/a[.//b][c//d]//e`)

Model evaluation:

- Use 5-fold cross-validation
- Plot predicted vs. actual costs
- Calculate accuracy measurements

Evaluating COMET's Accuracy

Error metrics:

- **NRMSE (Normalized Root-Mean-Squared Error)**: measures the average (relative) prediction error

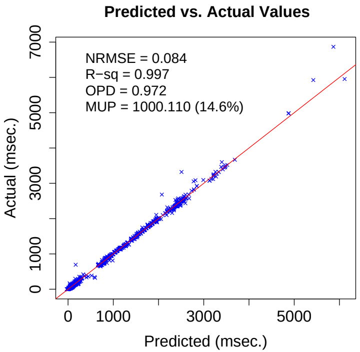
$$\text{NRMSE} = \frac{1}{\bar{c}}\left(\frac{1}{n}\sum_{i=1}^{n}\left(c_i - \hat{c}_i\right)^2\right)^{1/2}$$

  where c_i and $\hat{c}_i$ are the actual and estimated costs for ith query, and $\bar{c} = \text{average}(c_1, c_2, \ldots, c_n)$

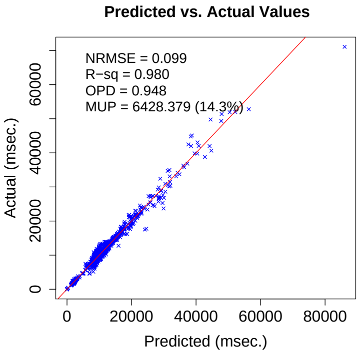
- Other metrics discussed in paper: $\mathbf{R^2}$, **OPD**, **MUP**

Accuracy of COMET

COMET does decent-to-excellent job in most cases:



(a) XMark (Mixed Queries)

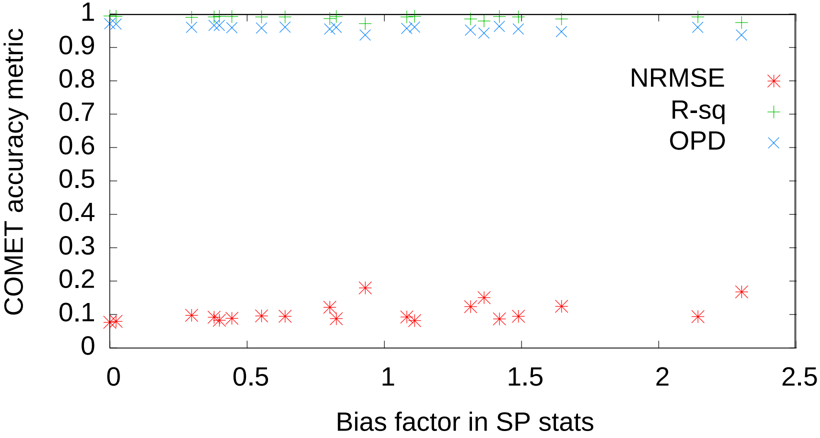


(b) TPC-H (Mixed Queries)

Add query type (simple, branching, complex) as feature?

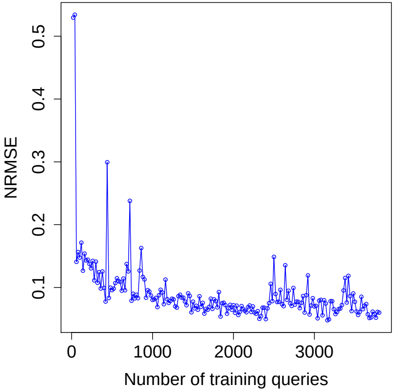
Effect of Errors in SP Statistics

COMET is not sensitive to systematic errors in SP stats:



Effect of Training-set Size

Training-set is of reasonable size for reasonable accuracy:



Model build time for
1000 training queries:
$<$ 1 second

Conclusion

Summary

- Statistical learning increasingly needed as data and its management become increasingly complicated
- COMET can accurately model XNAV cost
- COMET cost model is fast to construct and adaptable to changing environment
- A promising approach for costing complex query operators

Conclusion

Summary

- Statistical learning increasingly needed as data and its management become increasingly complicated
- COMET can accurately model XNAV cost
- COMET cost model is fast to construct and adaptable to changing environment
- A promising approach for costing complex query operators

Future Work

- Automatic identification of features
- Smarter generation of training queries
- Extensions to handle I/O costs, multi-user environments (will identify appropriate features)
- Incorporation of selectivity-estimation technology
- Improve dynamic model maintenance (incremental model building)
- Apply to other operators (XML, relational, text)