Xplus: A SQL-Tuning-Aware Query Optimizer

Herodotos Herodotou
Shivnath Babu
Duke University
**Motivation**

- Daily report generation workload
- Complex, long-running queries
- **Objective:** Finish by 6am

I need to improve performance of query Q by 5x

SQL Tuning

DBA

DB Server

reports

reports
Motivation

- Daily report generation workload
- Complex, long-running queries
- **Objective:** Finish by 6am

Can you generate a 5x better plan?

Xplus: a new optimizer that supports regular optimization & tuning sessions

9/16/2010

Duke University
Xplus: SQL-Tuning-Aware Optimizer

**Inputs**
1. Query $Q$ & plan $p$
2. Tuning Objectives
3. Resource Constraints

**Approach:** Proactively run (sub)plans for $Q$, & iterate based on the information collected from these runs

**Xplus Tuning Session**

**Estimated Cardinality (EC)**

**Actual Cardinality (AC)**

Table Scan ($R$) 1050
Table Scan ($S$) 850
Hash Join 700
Hash Join 900
Hash Join 100
Xplus: SQL-Tuning-Aware Optimizer

**Inputs**
1. Query $Q$ & plan $p$
2. Tuning Objectives
3. Resource Constraints

**Output**
1. Best plan $p'$ found so far
2. Performance improvement

Xplus Tuning Session

Cost of Best Plan vs Time
Xplus: SQL-Tuning-Aware Optimizer

**Inputs**
1. Query $Q$ & plan $p$
2. Tuning Objectives
3. Resource Constraints

**Optimality Guarantee**
Plan $p'$ is the *optimal plan* given
a) current database configuration
b) costing of plans using accurate cardinalities in optimizer cost model
Xplus: SQL-Tuning-Aware Optimizer

**Inputs**
1. Query $Q$ & plan $p$
2. Tuning Objectives
3. Resource Constraints

- **Main Challenges:**
  - Which (sub)plans to run to reach tuning goal efficiently?
  - Avoid/bound overhead on the production workload

Xplus Tuning Session

Cost of Best Plan vs. Time
Road Map

- Motivation
- Plan Space Representation
- Plan Space Search Strategy
- Implementation of Xplus
- Experimental Evaluation
New Representation of Plan Space

- Xplus views the physical plan space for a query as a collection of plan neighborhoods.
- **Goal:** Capture relationships among plans to make best use of information from (sub)plan runs.
- **Key enabler:** Cardinality set of an operator/plan/neighborhood.
Cardinality Sets

- To cost operator $O$:
  - Cardinality values
  - Cost formulas
- Cardinality Set of $O$
  - $\text{CS}(O) = \text{Set of relational algebra exprs whose cardinalities are used to cost } O$
- Cardinality Set of plan $p$
  - $\text{CS}(p) = \bigcup_{O \in p} \text{CS}(O)$

Diagram:

- $|\sigma_p(R) \bowtie S \bowtie \sigma_q(T)| = 30$
- $|\sigma_p(R) \bowtie S| = 100$
- $|\sigma_q(T)| = 80$
- $|\sigma_p(R)| = 700$
- $|S| = 850$
- $|T| = 200$
- $|R| = 2000$

$\text{CS}(H_1) = \{ \sigma_p(R), S, \sigma_p(R) \bowtie S \}$

$\text{CS}(p) = \{ R, S, T, \sigma_p(R), \sigma_q(T), \sigma_p(R) \bowtie S, \sigma_p(R) \bowtie S \bowtie \sigma_q(T) \}$
Plan Neighborhood

- **Partition plan space** based on cardinality set equality
  - $\text{CS}(p_i) = \text{CS}(p_j) \leftrightarrow p_i, p_j \text{ in Neighborhood } N$
  - Note: Xplus uses a richer definition of plan neighborhood based on set subsumption (*see paper*)

- **Intuition**: If accurate cardinality (AC) values to cost $p_i$ are available, then $p_j$ can be costed accurately as well

- **Cardinality Set of neighborhood** $N$
  - $\text{CS}(N) = \bigcup_{p \in N} \text{CS}(p)$
Coverage of Neighborhoods

- Neighborhood $N$ is covered when AC (accurate card.) values are available for all expressions in $\text{CS}(N)$
  - $N$ is covered all plans in $N$ are accurately costed
- Xplus progresses by covering more and more neighborhoods
Efficiency and Guarantees

- Run at most 1 plan in $N$ to obtain AC values for CS($N$)
- After a run, recosts minimal set of neighborhoods
- Cover all neighborhoods by running few subplans
- Covered space → **Optimality Guarantee** (Optimal plan found for given database configuration and cost model)
Road Map

- Motivation
- Plan Space Representation
- Plan Space Search Strategy
- Implementation of Xplus
- Experimental Evaluation
Challenges

- **Enumeration**
  - How to enumerate neighborhoods efficiently?
  - How to enumerate plans in a neighborhood?

- **Progress**
  - Which neighborhood to cover next?
  - Which (sub)plan to run to cover chosen neighborhood?
Neighborhood & Plan Enumeration

- **Transformations**: Functions applied to plan $p$ to produce valid plan $p'$
  - **Inter Transformations**
    - $p$ and $p'$ belong in different neighborhoods
  - **Intra Transformations**
    - $p$ and $p'$ belong in the same neighborhood

![Plan Space Diagram]

$N_1$, $N_2$, $N_3$, $N_4$ represent different neighborhoods in the plan space. $p_1$, $p_2$, $p_3$, $p_4$ are points in the plan space. The shaded areas represent uncovered regions, and the green areas represent covered regions.
Which Neighborhood to Cover Next?

- Suppose:
  - Plan \( p_i \) is least-cost plan in \( N_i \)
  - \( \text{Cost}(p_2) < \text{Cost}(p_3) < \text{Cost}(p_4) \)
    (based on uncertain cardinality values)
  - \( N_2, N_3 \) have 2 missing AC values, \( N_4 \) has 3
Which Neighborhood to Cover Next?

• Suppose:
  • Plan $p_i$ is least-cost plan in $N_i$
  • Cost($p_2$) < Cost($p_3$) < Cost($p_4$)
    (based on uncertain cardinality values)
  • $N_2, N_3$ have 2 missing AC values, $N_4$ has 3

• Options:
  • Pick $N_2$ → bring AC values faster
  • Pick $N_4$ → bring more AC values

• Solution:
  • Balance using goal-driven Experts & Selection Policy

Exploitation Vs Exploration
Exploitation Vs Exploration: Experts

Expert 1  Expert 2  Expert N

Selection Policy

<table>
<thead>
<tr>
<th>Expert</th>
<th>Description</th>
<th>Exploitation Component</th>
<th>Exploration Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pure Exploiter</td>
<td>Pick $N$ with least-cost plan</td>
<td>Highest</td>
<td>None</td>
</tr>
<tr>
<td>Join Shuffler</td>
<td>Focus on join selectivities</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Base Changer</td>
<td>Focus on the first two-way join</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Pure Explorer</td>
<td>Pick $N$ with most missing ACs</td>
<td>None</td>
<td>Highest</td>
</tr>
</tbody>
</table>
# Exploitation Vs Exploration: Policies

<table>
<thead>
<tr>
<th>Selection Policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round-Robin</td>
<td>Ensures fairness across experts</td>
</tr>
<tr>
<td>Priority</td>
<td>Realizes common approach: exploit before explore</td>
</tr>
<tr>
<td>Reward</td>
<td>Rewards based on past performance of experts</td>
</tr>
</tbody>
</table>
Which (Sub)Plan to Run to Cover $N$?

- **Goal**
  - Find *all missing* AC values in $CS(N)$ *as fast as possible*

- Run *least-cost, possibly modified*, plan in $N$

- **Plan-Modifications**
  - Subplan identification
  - Additional Scans

**Diagram:**
- **Hash Join**
  - $|σ_p(R) \bowtie S|$
- **INLJ**
  - $|S|$
- **σ_q**
  - **Table Scan (T)**
  - **Table Scan (S)**

**Missing AC values:** $\{ |S|, |σ_p(R) \bowtie S| \}$
Road Map

- Motivation
- Plan Space Representation
- Plan Space Search Strategy
- Implementation of Xplus
- Experimental Evaluation
Architecture

Global State Repository

Recommendation Engine
- Expert 1
- Expert N
- Selection Policy

Enumeration & Costing Engine

Controller

Plan Selector

Execution Agent
- Slot 1
- Slot M

9/16/2010
Duke University
Current Implementation of Xplus

- Usable as tuning tool
- Interacts with DB using standard interface
- No changes to DB execution engine

- Stand-alone Java app
- For PostgreSQL DB
Current Implementation of Xplus

- Extensible Components:
  - Experts
  - Policies
  - Controllers

- Controllers
  - Experts Controller (Serial / Parallel)
  - (IBM) Leo Controller
  - (Oracle) ATO Controller
Road Map

- Motivation
- Plan Space Representation
- Plan Space Search Strategy
- Implementation of Xplus
- Experimental Evaluation
Experimental Setup

- **Tuning Scenarios** (TPC-H Scale Factor 10)
  1. Query-level issues
  2. Data-level issues
  3. Statistics-level issues
  4. Physical-design issues

- **Evaluation Methodology**
  1. Overall performance of Xplus
  2. Comparison with other approaches: Leo, ATO
  3. Internal comparisons: selection policies and experts
## Overall Performance of Xplus

<table>
<thead>
<tr>
<th>Query</th>
<th>PostgreSQL Plan Run Time (sec)</th>
<th>Xplus Plan Run Time (sec)</th>
<th>Speedup Factor</th>
<th>Number of Subplans</th>
<th>Tuning Time Absolute (sec)</th>
<th>Tuning Time Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>257.55</td>
<td>21.38</td>
<td>12.0</td>
<td>6</td>
<td>131.58</td>
<td>0.51</td>
</tr>
<tr>
<td>2</td>
<td>8.67</td>
<td>0.59</td>
<td>14.8</td>
<td>5</td>
<td>40.42</td>
<td>4.66</td>
</tr>
<tr>
<td>5</td>
<td>1037.80</td>
<td>399.01</td>
<td>2.6</td>
<td>8</td>
<td>149.76</td>
<td>0.14</td>
</tr>
<tr>
<td>9</td>
<td>1722.27</td>
<td>754.82</td>
<td>2.3</td>
<td>8</td>
<td>870.78</td>
<td>0.51</td>
</tr>
<tr>
<td>11</td>
<td>20.00</td>
<td>3.55</td>
<td>5.6</td>
<td>2</td>
<td>29.11</td>
<td>1.46</td>
</tr>
<tr>
<td>16</td>
<td>15.90</td>
<td>0.77</td>
<td>20.7</td>
<td>2</td>
<td>27.04</td>
<td>1.70</td>
</tr>
<tr>
<td>20</td>
<td>3.36</td>
<td>2.32</td>
<td>1.4</td>
<td>4</td>
<td>7.13</td>
<td>2.13</td>
</tr>
<tr>
<td>21</td>
<td>509.51</td>
<td>72.17</td>
<td>7.1</td>
<td>4</td>
<td>45.83</td>
<td>0.09</td>
</tr>
<tr>
<td><strong>Totals:</strong></td>
<td><strong>97 min</strong></td>
<td><strong>32 min</strong></td>
<td><strong>3.0</strong></td>
<td></td>
<td><strong>24 min</strong></td>
<td><strong>0.25</strong></td>
</tr>
</tbody>
</table>

**Default Xplus Setting:** Parallel Experts Controller, 4 Experts, Priority Policy
Comparison With Other Approaches

<table>
<thead>
<tr>
<th>Query</th>
<th>Xplus</th>
<th>Leo Controller</th>
<th>ATO Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Speedup</td>
<td>Time</td>
</tr>
<tr>
<td>7</td>
<td>0.51</td>
<td>12.0</td>
<td>0.26</td>
</tr>
<tr>
<td>9</td>
<td>2.91</td>
<td>2.3</td>
<td>0.91</td>
</tr>
<tr>
<td>2</td>
<td>4.66</td>
<td>14.8</td>
<td>5.09</td>
</tr>
<tr>
<td>5</td>
<td>2.56</td>
<td>2.6</td>
<td>0.57</td>
</tr>
<tr>
<td>11</td>
<td>1.46</td>
<td>5.6</td>
<td>0.14</td>
</tr>
<tr>
<td>16</td>
<td>1.70</td>
<td>20.7</td>
<td>0.10</td>
</tr>
<tr>
<td>20</td>
<td>2.23</td>
<td>1.4</td>
<td>2.12</td>
</tr>
<tr>
<td>21</td>
<td>0.09</td>
<td>7.1</td>
<td>0.01</td>
</tr>
<tr>
<td><strong>Avg:</strong></td>
<td><strong>1.80</strong></td>
<td><strong>7.7</strong></td>
<td><strong>1.03</strong></td>
</tr>
</tbody>
</table>

**Task:** Find a plan that is 5x faster than the current plan
Evaluation of Policies and Experts

- **Selection Policy effects:**
  - Time to find better plan (convergence)
  - Time to find optimal plan (completion)
- **Exploitation-based policy**
  - Faster convergence
  - Longer completion
- **Exploration-based policy**
  - Longer convergence
  - Faster completion

![Graph showing convergence and completion times for different policies]
Summary

• Plan Space Representation
  • Cardinality Sets
  • Plan Neighborhood
• Plan Space Search Strategy
  • Exploitation Vs. Exploration
• Xplus
  • SQL-Tuning-Aware Query Optimizer

Thank You!
Other Approaches to SQL Tuning

- Query Execution Feedback
  - IBM’s Learning Optimizer (Leo)
  - MSR’s Pay-As-You-Go approach
  - Collect AC values during runs and relay to optimizer
- Oracle’s Automatic Tuning Optimizer (ATO)
  - Validate important estimates used by optimizer
- Adaptive Query Processing
  - Eddies
  - Rdb
Plan Neighborhood

• Exact Definition:
  • The space of physical plans for a query $Q$ can be represented as a graph $G_Q$. Each vertex in $G_Q$ denotes a physical plan for $Q$. An edge exists between the vertices for plans $p_1$ and $p_2$ if $CS(p_1) \subseteq CS(p_2)$ or $CS(p_2) \subseteq CS(p_1)$. The connected components of $G_Q$ define the plan neighborhoods of $Q$. 
Same Neighborhoods

$$|\sigma_p(R) \bowtie S \bowtie \sigma_q(T)|$$

$$|\sigma_p(R) \bowtie S|$$

$$|\sigma_q(T)|$$

$$|\sigma_p(R)|$$

$$|S|$$

$$|T|$$

$$\sigma_p$$

Table Scan ($S$)

Table Scan ($T$)

Hash Join

$$|\sigma_p(R) \bowtie S \bowtie \sigma_q(T)|$$

$$|\sigma_p(R) \bowtie S|$$

$$|\sigma_q(T)|$$

$$|\sigma_p(R)|$$

$$|S|$$

$$|T|$$

$$\sigma_p$$

Table Scan ($S$)

Table Scan ($T$)

Hash Join

$$|\sigma_p(R) \bowtie S \bowtie \sigma_q(T)|$$

$$|\sigma_p(R) \bowtie S|$$

$$|\sigma_q(T)|$$

$$|\sigma_p(R)|$$

$$|S|$$

$$|T|$$

$$\sigma_p$$

Table Scan ($S$)

Table Scan ($T$)

Hash Join

$$\text{CS}(p_1) = \{R, S, T, \sigma_p(R), \sigma_q(T), \sigma_p(R) \bowtie S, \sigma_p(R) \bowtie S \bowtie \sigma_q(T)\}$$

$$\text{CS}(p_2) = \{R, S, T, \sigma_p(R), \sigma_q(T), \sigma_p(R) \bowtie S, \sigma_p(R) \bowtie S \bowtie \sigma_q(T)\}$$
Different Neighborhoods

CS(p_1)=\{R, S, T, \sigma_p(R), \sigma_q(T), \\
\sigma_p(R) \bowtie S, \sigma_p(R) \bowtie S \bowtie \sigma_q(T)\}
\neq CS(p_3)=\{R, S, T, \sigma_p(R), \sigma_q(T), \\
\sigma_p(R) \bowtie \sigma_q(T), \sigma_p(R) \bowtie S \bowtie \sigma_q(T)\}
## Cardinality Sets for PostgreSQL Operators

<table>
<thead>
<tr>
<th>Physical Operator in PostgreSQL</th>
<th>Cardinality Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential Table Scan (TS) on table $S$</td>
<td>$S$</td>
</tr>
<tr>
<td>Index Scan (IS) on table $S$ with index predicate $p$</td>
<td>$\sigma_p(S)$</td>
</tr>
<tr>
<td>Bitmap Index Scan (BIS) on table $S$ with index predicate $p$</td>
<td>$S, \sigma_p(S)$</td>
</tr>
<tr>
<td>Bitmap AND/OR Operator over two or more Bitmap Index Scans</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Bitmap Heap Scan over expression $L$</td>
<td>$L$</td>
</tr>
<tr>
<td>Hash Join (HJ) over child expressions $L$ and $R$</td>
<td>$L, R, L \bowtie R$</td>
</tr>
<tr>
<td>Merge Join (MJ) over child expressions $L$ and $R$</td>
<td>$L, R, L \bowtie R$</td>
</tr>
<tr>
<td>Nested Loop Join (NLJ) over child expressions $L$ and $R$</td>
<td>$L, R, L \bowtie R$</td>
</tr>
<tr>
<td>Index Nested Loop Join (INLJ) over child expressions $L$ and $R$</td>
<td>$L, L \bowtie R$</td>
</tr>
<tr>
<td>Plain Aggregate (no grouping) over expression $L$</td>
<td>$L$</td>
</tr>
<tr>
<td>Hash/Sort-based GroupBy/Aggregate over expression $L$ with $L$ with grouping attribute $a$ and aggregate function $f$</td>
<td>$L, a G_f(L)$</td>
</tr>
<tr>
<td>Unique Operator over expression $L$ and attribute $a$</td>
<td>$L, a G(L)$</td>
</tr>
<tr>
<td>Sort Operator over expression $L$</td>
<td>$L$</td>
</tr>
<tr>
<td>Materialize Operator over expression $L$</td>
<td>$L$</td>
</tr>
</tbody>
</table>
### Progress of Execution Time for Q07

<table>
<thead>
<tr>
<th>Plan</th>
<th>Found by</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1p_1$</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>$N_1p_{1121}$</td>
<td>Pure Exploiter</td>
</tr>
<tr>
<td>$N_4p_{681}$</td>
<td>Join Shuffler</td>
</tr>
<tr>
<td>$N_8p_{1270}$</td>
<td>Base Changer</td>
</tr>
</tbody>
</table>

**Xplus Setting:** Serial Experts Controller, 4 Experts, Priority Policy