Plan Bouquets: A Fragrant Approach to Robust Query Processing

Jayant R. Haritsa
Database Systems Lab
Indian Institute of Science, Bangalore
Talk Theme

Declarative query processing with performance guarantees has been a highly desirable but equally elusive goal for the database community over the last three decades. I will present a conceptually new approach, called “plan bouquets”, to address this classical problem.

(Joint work with Anshuman Dutt, IISc PhD student)
Sample Relational Database: Manufacturing

- SQL query for
  - Complete details of orders for cheap parts
  - Algebraic equivalent: \( \sigma_{p\text{-retailprice} < 1000} (\text{part} \bowtie_{\text{partkey}} \text{lineitem} \bowtie_{\text{orderkey}} \text{orders}) \)
Query Execution Plan

• Ordered (imperative) sequence of steps to process the data

- Enormous number of semantically equivalent alternative plans
  - for a query with N relations, there are at least N! join orders
  - multiple algorithmic choices at each node in the plan
    (e.g. Join operator: Nested Loops, Sort Merge, Hash, Index, …)
Cost-based Query Optimization

- Determining the most efficient plan to execute an SQL query
  - Huge performance difference between good and bad plans
  - Only a few good plans
- Compare all alternative plans with a performance framework consisting of
  - **operator cardinality model**
    - estimate the quantity of data processing at each operator
    - expected to accurately estimate the number of tuples at each operator
      - summary statistics through histograms
  - **operator cost model**
    - estimate the time taken to perform the required data processing
    - expected to accurately estimate the
      - time taken to bring a relational page from disk to memory
      - time taken to process filter condition on a given tuple, etc.
Canonical Query Optimization Framework

Declarative Query (Q) → Query Optimizer → Optimal Plan P(Q) [Min Cost]

Operator Execution Cost Estimation Model

function of (Hardware, DB Engine)
e.g. NL Join
\[ = (|R_{outer}| + |R_{outer}| \times |R_{inner}|) \cdot t_{disk} \]

Operator Result Cardinality Estimation Model

function of (Data Distributions, Data Correlations)
e.g. Output Cardinality of Join
\[ = |t_{outer}| \times |t_{inner}| \times \text{join filter factor} \]
(EQ)

\[
\begin{align*}
\text{select} & \quad * \\
\text{from} & \quad \text{lineitem, orders, part} \\
\text{where} & \quad \text{p_partkey} = \text{l_partkey} \text{ and } \text{l_orderkey} = \text{o_orderkey} \text{ and } \text{p_retailprice} < 1000
\end{align*}
\]
Run-time Sub-optimality

The supposedly optimal plan-choice from the query optimizer may actually turn out to be highly sub-optimal when the query is executed with this plan. This adverse effect is due to errors in:

(a) cost model → limited impact, < 30 %
(b) cardinality model → huge impact, orders of magnitude

• Coarse statistics, outdated statistics, attribute value independence (AVI) assumption, multiplicative error propagation, query construction, …
The root of all evil, the Achilles Heel of query optimization, is the estimation of the size of intermediate results, known as cardinalities. Everything in cost estimation depends upon how many rows will be processed, so the entire cost model is predicated upon the cardinality model. In my experience, the cost model may introduce errors of at most 30% for a given cardinality, but the cardinality model can quite easily introduce errors of many orders of magnitude!
Cardinality Estimation Error (EQ)

```
select * 
from lineitem, orders, part 
where p_partkey = l_partkey 
    and l_orderkey = o_orderkey 
    and p_retailprice < 1000
```
Prior Research (lots!)

• **Sophisticated estimation techniques**

• **Selection of Robust Plans**

• **Runtime Re-optimization techniques**

Several novel ideas and formulations, but lacked performance guarantees
We present “plan bouquets”, a query processing technique that completely eschews making estimates for error-prone selectivities (normalized cardinalities [0% to 100%]).

Plan Bouquet Approach: run-time discovery of selectivities using a compile-time selected bouquet of plans
- provides worst case performance guarantees wrt omniscient oracle that knows the correct selectivities
  - e.g. for a single error-prone selectivity, relative guarantee of 4
- empirical performance well within guaranteed bounds on industrial-strength environments
Problem Framework
**Example Query: EQ**

```sql
select *
from lineitem, orders, part
where p_partkey = l_partkey and
  l_orderkey = o_orderkey and
  p_price < 1000
```

**SS – Selectivity Space**
Performance Metrics

- $q_e$ – estimated selectivity location in SS
- $q_a$ – actual run-time location in SS
- $P_{oe}$ – optimal plan for $q_e$
- $P_{oa}$ – optimal plan for $q_a$

$$SubOpt(q_e, q_a) = \frac{\text{cost}(P_{oe}, q_a)}{\text{cost}(P_{oa}, q_a)} \quad [1, \infty)$$

$$\text{MaxSubOpt (MSO)} = \text{MAX}[SubOpt(q_e, q_a)] \quad \forall q_e, q_a \in SS$$

$$\text{AvgSubOpt (ASO)} = \text{AVG}[SubOpt(q_e, q_a)] \quad \forall q_e, q_a \in SS$$
Main Assumptions

- Plan Cost Monotonicity (Mandatory)
- Perfect Cost Model (relaxed at end of talk)

PCM:

For any plan $P$ and distinct selectivity locations $q_1$ and $q_2$

$$\text{cost}(P, q_1) < \text{cost}(P, q_2) \quad \text{if} \quad q_1 < q_2$$

($q_1$ is dominated by $q_2$ in SS)
Current Optimizer Behavior on One-dimensional SS
Parametric Optimal Set of Plans (POSP)

(Parametric version of EQ)

\[
\begin{align*}
\text{select} & \quad * \\
\text{from} & \quad \text{lineitem, orders, part} \\
\text{where} & \quad \text{p\_partkey} = \text{l\_partkey} \text{ and} \\
& \quad \text{l\_orderkey} = \text{o\_orderkey} \text{ and} \\
& \quad \text{SEL (PART)} = $1
\end{align*}
\]

Using Selectivity Injection

NL: Nested Loop Join  L: Lineitem
MJ: Merge Join  O: Orders
HJ: Hash Join  P: Part
Sub-optimality Profile (across SS)

MaxSubOpt = 100

AvgSubOpt = 1.8

SubOpt (1%, 99%) = 20
Bouquet Approach in 1D SS
Bouquet Identification

Step 1: Draw cost steps with cost-ratio $r=2$ (geometric progression).

Step 2: Find plans at intersection of optimal profile with cost steps

Bouquet = \{P1, P2, P3, P5\}
Let $q_a = 5\%$

1. Execute $P_1$ with budget $IC_1(1.2E4)$
2. Throw away results of $P_1$ Execute $P_1$ with budget $IC_2(2.4E4)$
3. Throw away results of $P_1$ Execute $P_1$ with budget $IC_3(4.8E4)$
4. Throw away results of $P_1$ Execute $P_1$ with budget $IC_2(9.6E4)$
5. Throw away results of $P_1$ Execute $P_2$ with budget $IC_5(1.9E5)$
6. Throw away results of $P_2$ Execute $P_3$ with budget $IC_6(3.8E5)$

$P_3$ completes with cost $3.4E5$
Stupid Idea?

Yes! Extremely stupid!

We are expending lots and lots of wasted effort at both planning time (producing PIC) and at execution time (throwing away work)! Certainly a recipe for disaster ...

But, with careful engineering, can actually be made to work surprisingly well → rest of talk
Let \( q_a = 5\% \)

Bouquet Cost = \( 3.4 \text{ E5 (P3)} + 1.92 \text{ E5 (P2)} + 0.96 \text{ E5 (P1)} + 0.48 \text{ E5 (P1)} + 0.24 \text{ E5 (P1)} + 0.12 \text{ E5 (P1)} \)

\( = 7.1 \text{ E5} \)

SubOpt (*) \( q_a = 5\% \)

\( = 7.1/3.4 = 2.1 \)

With obvious optimization

SubOpt(*) \( q_a = 5\% \)

\( = 6.3/3.4 = 1.8 \)

P3 completes with cost \( 3.4 \text{E5} \)

Dec 2014

CMG Keynote
Bouquet Performance (EQ)

**Native Optimizer**
- MaxSubOpt = 100
- AvgSubOpt = 1.8

**Bouquet**
- MaxSubOpt = 3.1
- AvgSubOpt = 1.7
Worst Case Cost Analysis

Bouquet (upper bound)
Optimal (lower bound)

$P_k$ would complete within its budget when $q_a \in (q_{k-1}, q_k]$
1D Performance Bound

\[ C_{\text{bouquet}}(q_{k-1}, q_k) = \text{cost}(IC_1) + \text{cost}(IC_2) + \ldots + \text{cost}(IC_{k-1}) + \text{cost}(IC_k) \]

\[ = a + ar + \ldots + ar^{k-2} + ar^{k-1} \]

\[ = \frac{a(r^k - 1)}{(r - 1)} \]

\[ C_{\text{optimal}}(q_{k-1}, q_k) \geq ar^{k-2} \quad \text{(Implication of PCM)} \]

\[ \text{SubOpt}_{\text{bouquet}}(*, q_a) \leq \frac{1}{ar^{k-2}} \times \frac{a(r^k - 1)}{(r - 1)} \leq \frac{r^2}{r - 1} \quad \forall q_a \in (q_{k-1}, q_k) \]

Reaches minima at \( r = 2 \)

\[ \Rightarrow \text{MSO} = 4 \]

Best performance achievable by any deterministic online algorithm!
Connection to Online Bidding Problem

• There is an object $D$ with hidden value $V$ in range $(1, 100)$

• Your task is to bid for $D$ until you acquire it under the following rules:
  – If the bid $B < V$, then you forfeit $B$, and bid again
  – If the bid $B \geq V$, then you pay $B$ and acquire $D$

• Your goal is to minimize the worst-case ratio of your total payment to the object value, i.e.
  $$\min \left( \frac{B_1 + B_2 + \ldots + B_k}{V} \right)$$

• Bid doubling algorithm is best possible choice
Bouquet Approach in 2D SS
2D Bouquet Identification

- Cost (normalized)
- Plans
- Isocost Planes
- Multiple Plans per contour

sel-X

sel-Y

Dec 2014 CMG Keynote
Characteristics of 2D Contours

- Hyperbolic curves
- Multiple plans per contour

Third quadrant coverage (due to PCM)
P$_2^k$ can complete any query with actual selectivities(q$_a$) in the shaded region within cost(IC$_k$)
Crossing 2D Contours

- Covered by all plans in contour
- Covered by only one plan in contour

⇒ Entire set of contour plans must be executed to fully cover all locations under \( IC_k \)
2D Performance Analysis

- When $q_a \in (IC_{k-1}, IC_k]$

$$C_{bouquet}(q_a) = \sum_{i=1}^{k} [n_i \times \text{cost}(IC_i)]$$

Number of plans on $i^{th}$ contour

$$\rho = \max(n_i)$$

$$C_{bouquet}(q_a) \leq \rho \times \sum_{i=1}^{k} \text{cost}(IC_i)$$

$$SubOpt_{bouquet}(q_a) = 4\rho$$  (Using 1D Analysis)

Bound for N-dimensions:  $$SubOpt_{bouquet}(q_a) = 4 \times \rho_{IC_{surface}}$$
Dealing with large $\rho$

- In practice, $\rho$ can often be large, even in 100s, making the performance guarantee of $4\rho$ impractically weak

- Reducing $\rho$
  - Compile Time:
    - Anorexic POSP reduction [VLDB 2007]
  - Run Time:
    - Explicit Monitoring of Selectivity Lower Bounds
    - Spilling-based Execution
1) Reducing $\rho$ with Anorexic Reduction

- Collapse a large set of POSP plans on a selectivity space into a reduced cover that provides performance within a $(1+\lambda)$ factor of the optimal at all locations in the ESS. With $\lambda = 0.2$, invariably obtain a small-sized (< 10) cover.
### MSO guarantees (compile time)

<table>
<thead>
<tr>
<th>Query (dim)</th>
<th>$\rho_{\text{POSP}}$</th>
<th>MSO Bound (POSP) $= 4\rho_{\text{POSP}}$</th>
<th>$\rho_{\text{reduced}}$ ($\lambda=0.2$)</th>
<th>MSO Bound (reduced) $= 4\rho_{\text{reduced}}(1+\lambda)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q5 (3D)</td>
<td>11</td>
<td>44</td>
<td>3</td>
<td>14.4</td>
</tr>
<tr>
<td>Q7 (3D)</td>
<td>13</td>
<td>52</td>
<td>3</td>
<td>14.4</td>
</tr>
<tr>
<td>Q8 (4D)</td>
<td>88</td>
<td>352</td>
<td>7</td>
<td>33.6</td>
</tr>
<tr>
<td>Q7 (5D)</td>
<td>111</td>
<td>444</td>
<td>9</td>
<td>43.2</td>
</tr>
<tr>
<td>Q15 (3D)</td>
<td>7</td>
<td>28</td>
<td>3</td>
<td>14.4</td>
</tr>
<tr>
<td>Q96 (3D)</td>
<td>6</td>
<td>24</td>
<td>3</td>
<td>14.4</td>
</tr>
<tr>
<td>Q7 (4D)</td>
<td>29</td>
<td>116</td>
<td>4</td>
<td>19.2</td>
</tr>
<tr>
<td>Q19 (5D)</td>
<td>159</td>
<td>636</td>
<td>8</td>
<td>38.4</td>
</tr>
<tr>
<td>Q26 (4D)</td>
<td>25</td>
<td>100</td>
<td>5</td>
<td>24.0</td>
</tr>
<tr>
<td>Q91 (4D)</td>
<td>94</td>
<td>376</td>
<td>9</td>
<td>43.2</td>
</tr>
</tbody>
</table>
Bouquet Architecture

Identification of error-prone selectivities

Query Optimizer

Statistics

Data

Query Executor
Empirical Evaluation
Experimental Testbed

- Database Systems: PostgreSQL and COM (commercial engine)
- Databases: TPC-H and TPC-DS
- Physical Schema: Indexes on all attributes present in query predicates

- Workload: 10 complex queries from TPC-H and TPC-DS
  - with SS having up to 5 error dimensions

- Metrics: Computed MSO and ASO using Abstract Plan Costing over SS
Performance on PostgreSQL

• For many DS queries
  – MSO improves from $\approx 10^6$ to $\approx 10$
  – ASO improves from $\approx 10^2$ to $\approx 5$

ASO not compromised to reduce MSO!
Performance with COM

⇒ Robustness improvements not artifact of a specific engine
Incorporating Cost Model Error

- If cost model error is bounded by $\delta$, that is
  \[
  \frac{\text{cost}_{\text{estimated}}}{\text{cost}_{\text{actual}}} \in \left[\frac{1}{1+\delta}, 1 + \delta\right]
  \]
  then
  \[
  MSO_{\text{bounded}} \leq MSO_{\text{perfect}} \times (1 + \delta)^2
  \]
  
  - for $\delta = 0.4$ \(\Rightarrow\) \(MSO_{\text{bounded}} \leq 2 \cdot MSO_{\text{perfect}}\)
Summary

• Plan bouquet approach achieves
  – bounded performance sub-optimality
    • using a (cost-limited) plan execution sequence guided by isocost contours defined over the optimal performance curve
  – robust to changes in data distribution
    • only $q_a$ changes – bouquet remains same
  – easy to deploy
    • bouquet layer on top of the database engine
  – repeatability in execution strategy (important for industry)
    • $q_e$ is always zero, depends only on $q_a$
    • independent of metadata contents

Important distinction from re-optimization techniques
For more details, visit project website:

dsl.serc.iisc.ernet.in/projects/QUEST

• Concepts paper: ACM Sigmod 2014
• Demo paper: VLDB 2014
Take Away

Plan 1
Plan 2
Plan 3
Plan 4
Plan 5

SQL Query

Near Optimal Query Execution Performance

Do you know the correct selectivities?