Augmenting the Planner with Machine Learning

Manchester PostgreSQL Meetup
Anthony Kleerekoper
Example Query

SELECT
    sum(l_extendedprice * l_discount) as revenue
FROM
    lineitem
WHERE
    l_shipdate >= date '1997-01-01'
    and l_shipdate < date '1997-01-01' + interval '1' year
    and l_discount between 0.07 - 0.01 and 0.07 + 0.01
    and l_quantity < 25
LIMIT 1;
Example Query Plan

**QUERY PLAN**

```
Limit (cost=162260.44..162260.45 rows=1 width=12)

  -> Aggregate (cost=162260.44..162260.45 rows=1 width=12)

    -> Bitmap Heap Scan on lineitem
        (cost=19396.94..161661.72 rows=119743 width=12)

        -> Bitmap Index Scan on l_shipdate_idx
            (cost=0.00..19367.00 rows=923457 width=0)
```
Example Query Plan

QUERY PLAN

Limit (cost=162260.44..162260.45 rows=1 width=12)

-> Aggregate (cost=162260.44..162260.45 rows=1 width=12)

-> Bitmap Heap Scan on lineitem (cost=19396.94..161661.72 rows=119743 width=12)

-> Bitmap Index Scan on l_shipdate_idx (cost=0.00..19367.00 rows=923457 width=0)

How long is that?
Agenda

- High Level Overview of the Planner
  - What does the planner do?
- Query Performance Prediction
  - How long will this query take?
- Offloading the Executor
  - Can I run this join faster on a GPU?
- Augmenting the Planner
  - When is GPU faster?
“Crazy” Improvements

From “Inside the PostgreSQL Query Optimizer”, Neil Conway, 2005

- “Online statistics gathering”
- “Executor → optimizer online feedback”
- “Parallel query processing on a single machine (one query on multiple CPUs concurrently)”
Who am I?

- Research Associate at University of Manchester
- Background more in Machine Learning
- Working on PostgreSQL for about 18 months

anthonykleerekoper@gmail.com
High Level View of PostgreSQL

From “A Tour of PostgreSQL Internals” by Tom Lane
What does the Planner Do?

- Ideally – find fastest possible method of executing the query
- Actually – find the fastest method in a reasonable time
- Two steps:
  - Create plan or partial plan
  - Decide how long it will take
Lots of Choices

- **Scans:**
  - Sequential, index only, bitmap index

- **Joins:**
  - Nested loop, hash, merge

- **Aggregate:**
  - Hash or sort
"System R Algorithm"

- IBM System R dates to the 1970s
- Planning method based on method described in 1979 paper

Finding Join Paths

Consider all possible join orders?
- $n$ relations $\rightarrow n!$ Permutations
- $k+1^{\text{st}}$ join is independent of order of first $k$

Therefore:
- Try all possible orderings for 2 relations
- Keep the “good” ones, throw away the rest
- For $3^{\text{rd}}$ relation, consider possible orderings with best from before
- etc ...
What is “Good”? 

- Fastest to execute 
- Must estimate execution time → Cost 
- Cost is an analytical model with some guesswork 
  - Assume disk I/O dominates 
  - Assume ratios between operations 
  - Approximate selectivity of operations
## PostgreSQL Cost Model

<table>
<thead>
<tr>
<th>Cost Variable</th>
<th>Symbol</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>seq_page_cost</td>
<td>$c_s$</td>
<td>1.0</td>
</tr>
<tr>
<td>random_page_cost</td>
<td>$c_r$</td>
<td>4.0</td>
</tr>
<tr>
<td>cpu_tuple_cost</td>
<td>$c_t$</td>
<td>0.01</td>
</tr>
<tr>
<td>cpu_index_tuple_cost</td>
<td>$c_i$</td>
<td>0.005</td>
</tr>
<tr>
<td>cpu_operator_cost</td>
<td>$c_o$</td>
<td>0.00025</td>
</tr>
</tbody>
</table>

$$ C_O = n^Tc = n_sc_s + n_rc_r + n_tc_t + n_ic_i + n_oc_o $$
Example of Cost in Action

- TPC-H table “customer”
- 300,000 rows
  - key ranging from 0 to 149,999
  - then repeat again

```sql
EXPLAIN SELECT * FROM customer
WHERE c_custkey > cutoff;
```
# Example of Cost in Action

<table>
<thead>
<tr>
<th>Cutoff</th>
<th>True Count</th>
<th>Estimated Rows</th>
<th>Planner Cost</th>
<th>Proportion of Table</th>
<th>Chosen Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>280,000</td>
<td>280,002</td>
<td>11138.00</td>
<td>93.33%</td>
<td>Seq Scan</td>
</tr>
<tr>
<td>75,000</td>
<td>150,000</td>
<td>150,176</td>
<td>11138.00</td>
<td>50.00%</td>
<td>Seq Scan</td>
</tr>
<tr>
<td>90,000</td>
<td>120,000</td>
<td>120,296</td>
<td>11138.00</td>
<td>40.00%</td>
<td>Seq Scan</td>
</tr>
<tr>
<td>91,500</td>
<td>117,000</td>
<td>117,099</td>
<td>11138.00</td>
<td>39.00%</td>
<td>Seq Scan</td>
</tr>
<tr>
<td>92,000</td>
<td>116,000</td>
<td>116,041</td>
<td>11018.25</td>
<td>38.66%</td>
<td>Bitmap Heap Scan</td>
</tr>
<tr>
<td>95,000</td>
<td>110,000</td>
<td>109,930</td>
<td>10826.50</td>
<td>36.66%</td>
<td>Bitmap Heap Scan</td>
</tr>
<tr>
<td>100,000</td>
<td>100,000</td>
<td>99,579</td>
<td>10500.90</td>
<td>33.33%</td>
<td>Bitmap Heap Scan</td>
</tr>
<tr>
<td>125,000</td>
<td>50,000</td>
<td>50,206</td>
<td>8961.09</td>
<td>16.66%</td>
<td>Bitmap Heap Scan</td>
</tr>
<tr>
<td>140,000</td>
<td>20,000</td>
<td>19,794</td>
<td>8009.25</td>
<td>6.66%</td>
<td>Bitmap Heap Scan</td>
</tr>
<tr>
<td>145,000</td>
<td>10,000</td>
<td>9,675</td>
<td>8087.57</td>
<td>3.33%</td>
<td>Bitmap Heap Scan</td>
</tr>
<tr>
<td>149,000</td>
<td>2,000</td>
<td>2,020</td>
<td>4560.62</td>
<td>0.66%</td>
<td>Bitmap Heap Scan</td>
</tr>
<tr>
<td>149,500</td>
<td>1,000</td>
<td>1,010</td>
<td>2801.34</td>
<td>0.33%</td>
<td>Bitmap Heap Scan</td>
</tr>
<tr>
<td>149,600</td>
<td>800</td>
<td>808</td>
<td>2325.95</td>
<td>0.27%</td>
<td>Index Scan</td>
</tr>
<tr>
<td>149,900</td>
<td>200</td>
<td>202</td>
<td>608.04</td>
<td>0.07%</td>
<td>Index Scan</td>
</tr>
</tbody>
</table>
Agenda

- High Level Overview of the Planner
  - What does the planner do?
- Query Performance Prediction
  - How long will this query take?
- Offloading the Executor
  - Can I run this join faster on a GPU?
- Augmenting the Planner
  - When is GPU faster?
Query Performance Prediction

- We want to know, in advance, how long a query is going to take to run

Options:
- Rely on the planner's cost
- "Fix" the planner's cost
- Ignore the planner's cost
- Post-process the cost

PostQuePP – PostgreSQL Query Performance Prediction plugin
Can we rely on the planner cost?

- Use EXPLAIN – get cost of a query
- Does Cost = Execution Time?
Can we rely on the planner cost?

- Use EXPLAIN – get cost of a query
- Does Cost = Execution Time?

“The most critical part of the display is the estimated statement execution cost, which is the planner’s guess at how long it will take to run the statement (measured in cost units that are arbitrary, but conventionally mean disk page fetches)"

PostgreSQL 9.4.2 Documentation on EXPLAIN
Can we rely on the planner cost?
Can we “fix” the planner's cost?

\[ C_O = n^T c = n_s c_s + n_r c_r + n_t c_t + n_i c_i + n_o c_o \]

- Calibrate the constants
  - Create set of queries with known cardinalities
  - Solve set of simultaneous equations

\[
N = \begin{pmatrix}
0 & 0 & n_{t1} & 0 & 0 \\
0 & 0 & n_{t2} & 0 & n_{o2} \\
0 & 0 & n_{t3} & n_{i3} & n_{o3} \\
n_{s4} & 0 & n_{t4} & 0 & 0 \\
n_{s5} & n_{r5} & n_{t5} & n_{i5} & n_{o5}
\end{pmatrix}
\]
Can we “fix” the planner's cost?

\[ C_O = n^T c = n_s c_s + n_r c_r + n_t c_t + n_i c_i + n_o c_o \]

- Improve cardinality estimates
  - Extra sampling for chosen plan
  - Overhead of between 4% and 20%
  - Only usable for selections and joins

Can we “fix” the planner's cost?

Sample result:
Workload v Ad Hoc

- Have we seen similar queries before?
- Were they run on similar data?
- If yes → workload queries, “business intelligence”
- If not → ad hoc queries

- Machine Learning works best for workloads
What is Machine Learning?

- Automatically notice general patterns in data
- Construct a model that maps from data to class label or target
A Simple Example

- There is some suggestion that taller people get paid more than shorter people
  - Possibly even “per inch” bonus
- “How much is income influenced by height and sex?” by Stephen L. Brown, PhD
  - On website shortsupport.com
  - “These represent a 2.55% per inch premium for females and a startling (at least to me) 3.47% per inch premium for males.”
- Assume true → can we predict your pay from your height?
How Much per Inch?

- First gather “training” data
  - Records lots of data about a cohort, started in 1979 and in 1997

- Organise our data:
  - Feature is height
  - Target is income
Building a Model

- What kind of relationship might exist between height and pay?
- Assume linear:
  - Pay = Height x Bonus + BaseLine
- Use training data to find the “correct” values for “Bonus” and “BaseLine”
- Many methods to do this
Maybe it's not Linear?

- Could be quadratic?
  - Pay = $\text{Height}^2 \times \text{Bonus} + \text{BaseLine}$

- Could be irregular?
  - Pay = \text{Base} if Height < 6'
  - Pay = \text{Base} \times 2.5 if Height > 6'

- Learning is as good as the model used
- No “one size fits all”
Maybe Height is Irrelevant?

- Other factors affecting pay
- Perhaps cannot predict from height alone
- Need to select the right features
- “Garbage in, garbage out”
Seems Height is Irrelevant
Nearest Neighbour Regression

- Assume that similar examples have similar targets
  - Houses of similar size have similar prices
- For some definition of “similar”
  - Similar size AND location
- Find $k$ nearest neighbours in training data
- Average their targets
  - Perhaps add some weighting
Should we Ignore the Planner's Cost?

Can extract information direct from the plan

**QUERY PLAN**

Limit (cost=162260.44..162260.45 rows=1 width=12)

  -> Aggregate (cost=162260.44..162260.45 rows=1 width=12)

     -> Bitmap Heap Scan on lineitem
        (cost=19396.94..161661.72 rows=119743 width=12)

        -> Bitmap Index Scan on l_shipdate_idx
           (cost=0.00..19367.00 rows=923457 width=0)
Alternative Features from Plan

```
SELECT n.name
FROM tbNation n, tbRegion r
WHERE n.region = r.region
AND r.name = 'EUROPE'
ORDER BY n.name;
```
Kernel Canonical Component Analysis

Very Good for Workloads

![Graph showing the relationship between KCCA Predicted Elapsed Time and Actual Elapsed Time. The graph includes points indicating under-estimated records accessed, and a disk I/O estimate too high.](image)
Pretty Poor for Ad Hoc Queries

The graph illustrates the comparison between actual elapsed time and KCCA predicted elapsed time. The data points are aligned closely with the line of perfect prediction, indicating that the KCCA model's predictions are very close to the actual times for different time intervals (1 second, 1 minute, 1 hour).
Post-Process Planner Cost

- Take the Cost as the feature and build a model around it
- Perhaps a linear model?

Post-Process Planner Cost

- Take the Cost as the feature and build a model around it
- Perhaps a linear model?
- Perhaps not

What about Nearest Neighbour?

- Cost is a complex model
- Take Cost as “value” of a plan
- Nearest neighbour is plan with closest cost

<table>
<thead>
<tr>
<th>k</th>
<th>This Approach</th>
<th>KCCA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Uniform</td>
<td>Skewed</td>
</tr>
<tr>
<td>3</td>
<td>1.31%</td>
<td>6.44%</td>
</tr>
<tr>
<td>5</td>
<td>1.50%</td>
<td>6.63%</td>
</tr>
<tr>
<td>7</td>
<td>1.63%</td>
<td>6.49%</td>
</tr>
<tr>
<td>9</td>
<td>1.78%</td>
<td>6.31%</td>
</tr>
</tbody>
</table>
Advantages

- Much less data to store
  - One number per example
  - Two numbers per possible operator
- Much faster
  - Logarithmic time to find nearest neighbours
  - Exponential training time for KCCA
Advantages

- Exponential training time for KCCA
Disadvantages

- No good for Ad Hoc Queries
PostQuePP

- Integrate this into PostgreSQL
- PostgreSQL Query Performance Prediction plugin
- Make use of `planner_hook`
  - Wraps around planning process
  - Does not affect planning
- Can output prediction to users
  - Change cost inside the plan?
  - Confidence measures as well?
PostQuePP Feedback

- Continue to gather training data online
- Use `ExecutorEnd_hook`
- May need to age training data?
- Two “crazy” improvements:
  - “Online statistics gathering”
  - “Executor → optimizer online feedback”
PostQuePP Plugins

- ExecutorEnd_hook
- Feedback plugin
- Training Data
- planner_hook
- PostQuePP plugin
Agenda

- High Level Overview of the Planner
  - What does the planner do?
- Query Performance Prediction
  - How long will this query take?
- Offloading the Executor
  - Can I run this join faster on a GPU?
- Augmenting the Planner
  - When is GPU faster?
Two Types of Parallelism

- Can run multiple queries concurrently
- Execute parts of the same query in parallel
- Third “crazy” improvement:
  - “Parallel query processing on a single machine (one query on multiple CPUs concurrently)”
- BUT using GPUs or FPGAs
CPUs and GPUs and FPGAs! Oh My!

Scalar processing

SIMD processing (e.g. GPU)

Dataflow processing / pipelining (FPGAs)

Pipelining often better for global states

Slide from Dirk Koch
Different Ways of Parallelising

- CPU  a single worker
- SIMD multiple, independent workers
- GPU  one worker per problem
  - Synchronisation issues
- FPGA one worker per part
  - Free synchronisation
Bitonic Merge Sort

- Sorting can be an expensive operation
- Bitonic Merge Sort is well suited for parallelism
OpenCL and Just-in-Time

- C-like programming language for GPUs etc
- Just-in-Time compilation allows us to write general code
- Run whole or part of queries on GPUs, Accelerators, FPGAs
Agenda

- High Level Overview of the Planner
  - What does the planner do?
- Query Performance Prediction
  - How long will this query take?
- Offloading the Executor
  - Can I run this join faster on a GPU?
- Augmenting the Planner
  - When is GPU faster?
When to Use the GPU?

- Sometimes GPUs are much faster
- Sometimes much slower
- Develop a cost model?
  - Very variable
  - How to compare to disk I/O?
How Good is the Planner?

“Query Planning Gone Wrong”, Robert Haas

Looked at 168 complaints on PostgreSQL performance

~50% complains because of planner error

~1/3 because best plan was rejected
  ~30% row count errors
  ~5% cost errors

How many others not reported?
A New Way of Costing

- Cost model is Machine Learning without the learning
- Machine Learning already proved itself, in some cases at least
- Help the planner with Machine Learning
  - Replace the fixed cost with a learned one
Move From This ...

- SQL Query
- Parser
- Planner
  - Seq Scan
  - Index Scan
  - Bitmap Scan
  - Cost
  - Cost
  - Cost
- Cheapest
Cost Hooks and Plugins

- Add new hooks to PostgreSQL
- Place in `src/backend/optimizer/path/costsize.c`
- One hook per operator
  - Can be used by anyone to augment planner
  - Or change analytical cost model
- Plugins to use hooks to predict execution times
Some Early Results

![Graph showing median execution time for queries]

- **Original Execution Time**
- **ML Execution Time**
Still Challenges

- Choose best model
  - Maybe different models for different nodes
- Good coverage
  - Must train using even really bad plans
- Needs feedback
  - Learn from bad decisions
- Mix learned with analytical model?
Summary

- Planning is hard
- Machine Learning can help
- Adding ML infrastructure to PostgreSQL
- Early results are positive
Thanks for listening

The research leading to these results has received funding from the European Union’s Seventh Framework Programme (FP7/2007-2013) under grant agreement number 318633.