Batch Query Memory Prediction Using Deep Query Template Representations

GRADUATE PROGRAM IN ELECTRICAL AND COMPUTER ENGINEERING

SUPERVISORS: MANOS PAPAGELIS, MARIN LITOIU

M.S.C. THESIS OF NICOLAS ANDRES JARAMILLO DURAN
Motivation
DBMS – Query Optimization

Motivation
Problem Definition
Methodology
Experimental Evaluation
Contribution & Future Work

SELECT Q9.NAME AS "COMPANY_NAME",
FROM AHNAF.TITLE AS Q1,
AHNAF.MOVIE_LINK AS Q2,
AHNAF.MOVIE_KEYWORD AS Q3,
WHERE (Q1.MOVIE_ID = Q2.MOVIE_ID) AND (Q3.MOVIE_ID = Q2.MOVIE_ID) AND (Q1.MOVIE_ID = Q3.MOVIE_ID)
Towards an Autonomous DBMS

Physical Design

Which index should I create?

Query Performance Prediction

How long will it take to process the query at hand?

Configuration Tuning

What are the optimum configurations for my database?

Resource Forecasting

What resources are needed to process this query?

Motivation Problem Definition Methodology Experimental Evaluation Contribution & Future Work

Nicolas Andres Jaramillo Duran
Resource Forecasting

- Resources
  - Memory
  - Disk I/O
  - Throughput

- Challenging problem
  - Excessive overhead
  - Inefficient scheduling
  - Inappropriate allocation of resources

Motivation
Problem Definition
Methodology
Experimental Evaluation
Contribution & Future Work
Single Query vs Workload (Batch Queries)

- **Single Query**
  - \( x_1, x_2, \ldots, x_n \)

- **Workload (Batch Queries)**
  - \( \{ x_1, x_2, \ldots, x_n \} \)
  - Size \( s=5 \)

---

**Motivation**  |  **Problem Definition**  |  **Methodology**  |  **Experimental Evaluation**  |  **Contribution & Future Work**
---|---|---|---|---
6  |  |  |  |  

Nicolas Andres Jaramillo Duran
can we design **machine learning** models for predicting **memory** resource forecasting for a **batch of queries**?
Problem Definition
**Problem Definition**

**Input**
A query workload of size \(k\)

**Output**
Query Workload memory consumption estimate:

\[
\begin{align*}
y_1 \\
y_2 \\
\vdots \\
y_n
\end{align*}
\]
Problem Definition

We formulate the estimation of memory usage for an unseen workload as a **distribution regression problem**

**Problem:**

Given

- \( w_i = (Q_i, y_i) \)
- A training corpus \( \{(Q_1, y_1), \ldots, (Q_n, y_n)\} \)

We wish to learn a function, \( \hat{f}(\cdot) \), an approximation of \( f(\cdot) \)

\[ \hat{f}(\cdot), \text{can compute } \hat{y}, \text{an accurate estimate of the actual memory usage } y \]

\[ \hat{f}(w) = \hat{y} \]

**Assumptions:**

- An underlying function, \( f(\cdot) \)

\[ f(w) = y \]
Methodology
Three-Step Approach for Batch Query Memory Prediction

Phase 1
Learning Query Templates

Phase 2
Constructing Histograms from Workloads

Phase 3
Training a Distribution Regression Machine Learning Model

Motivation  Problem Definition  Methodology  Experimental Evaluation  Contribution & Future Work
Phase 1: Learning Query Templates

Motivation

Problem Definition

Methodology

Experimental Evaluation

Contribution & Future Work
Phase 1: Learning Query Templates

**Rule Base**
- Map the generated queries back to their corresponding templates
- Group queries based on their estimated cardinality similarity

**Clustering Base**
- **K-Means**
  - Query text feature encoding
    - Bag of words (BoW)
  - Query plan feature encoding
    - Cardinality cost aggregation for each operator (CCAEO)
    - Query plan encoding while maintaining tree structure (QPEWMTS)

- **K-meadois**
  - Query text feature encoding
    - Bag of words (BoW)
  - Query plan feature encoding
    - Cardinality cost aggregation for each operator (CCAEO)
    - Query plan encoding while maintaining tree structure (QPEWMTS)

- **DBSCAN**
  - Query text feature encoding
    - Bag of words (BoW)
  - Query plan feature encoding
    - Cardinality cost aggregation for each operator (CCAEO)
    - Query plan encoding while maintaining tree structure (QPEWMTS)
Phase 1: Learning Query Templates through CCAEO and K-Means

Problem Definition

Motivation

Methodology

Experimental Evaluation

Contribution & Future Work

Nicolas Andres Jaramillo Duran
Phase 2: Constructing Histograms from Workloads

### Constructing the workload

- **Logs**
- **Execution Engine**

### Constructing the Histogram

- $w_1, w_2, \ldots, w_n$
- $y_i = \sum_{j=1}^{[Q]} m_j$ aggregated memory cost

### Methodology

1. **Problem Definition**
2. **Motivation**
3. **Contribution & Future Work**
4. **Experimental Evaluation**
5. **Methodology**
6. **Phase 2: Constructing Histograms from Workloads**

---

Nicolas Andres Jaramillo Duran
Phase 3: Training the Model

- Ridge
- Decision Tree
- Random Forest
- XGBoost
- Deep Neural Network
Phase 3: Training a Distribution Regression Deep Learning Model

**Motivation**

**Problem Definition**

**Methodology**

**Experimental Evaluation**

**Contribution & Future Work**
Training Pipeline Overview

Phase 1: Learning Query Templates
- TR1: Extract query plans, memory usage
- TR2: Generate plan features
- TR3: Learn templates

Phase 2: Construct Histograms
- TR4: Create workload batches
  \[ w_i = q_1, q_2, ..., q_{|Q|} \]
  \[ m_1, m_2, ..., m_{|Q|} \]
  \[ y_i = \sum_{j=1}^{|Q|} m_j \] aggregated memory cost

\[ \{(w_1, y_1), ..., (w_i, y_i), ..., (w_n, y_n)\} \]

Phase 3: Training Model
- TR6: Train Model

Trained Model

Motivation Problem Definition Methodology Experimental Evaluation Contribution & Future Work
Inferencing Pipeline Overview

**Environment**
- Users
- Applications
- Query
- Response
- Database

**Phase 1: Learning Query Templates**
- IN1: Extract query plans
- IN2: Generate plan features
- IN3: Look up templates

**Phase 2: Construct Histograms**
- IN4: Generating distribution of templates
- IN5: Predict memory demand

**Phase 3: Training Model**
- Execution Engine

**Motivation**
- Problem Definition
- Methodology
- Experimental Evaluation
- Contribution & Future Work
Experimental Evaluation
Enviroment

- DB2 instance
- Linux system
- 8 CPU cores
- 32 GB of memory
- 500 GB of disk space
Datasets

TPCDS
- 93,000 queries
- OLAP – Transactional Workload

TPCC
- 3958 queries
- OLTP – Analytical Workload

JOB
- 2300 queries
- Join benchmark
- OLAP Transactional Workload
Experimental Evaluation Goals

Q1) Templates
• Learning query templates performance

Q2) LearnedWMP
• LearnedWMP accuracy performance
• LearnedWMP training and inference runtime cost
• LearnedWMP model size

Q3) Parameter
• Effect of the batch size parameter s
Q1 - Learning Query Templates Experimental Setup

Training Set

Logs

Qtrain

Execution Engine

TPC-DS

Phase 1 Learning Query Templates

K-Means

K-meadois

DBSCAN

Rule Base

CCAEO

QPEWMTS

Bag or words

Word Embedding

Text Mining

Phase 2 Construct Histograms

TR4 Create workload batches

\[ m_1, m_2, ..., m_{|Q|} \]

\[ y_i = \sum_{j=1}^{q_i} m_j \]

aggregated memory cost

\[ \{(w_1, y_1), ... (w_i, y_i), ... (w_n, y_n)\} \]

Phase 3 Training Model

TR6 Train XGBoost Model

XGBoost

Motivation

Problem Definition

Methodology

Experimental Evaluation

Contribution & Future Work
Q1 - Learning Query Templates Performance on TPC-DS Dataset

K-means

DBSCAN

K-medoids

Rule Base
Q1 - Learning Query Templates Performance on JOB Dataset

K-means

DBSCAN

Rule Base

Motivation | Problem Definition | Methodology | Experimental Evaluation | Contribution & Future Work
Q2 - LearnedWMP Experimental Setup

LearnedWMP

Machine Learning
SingleWMP

Non-Machine Learning
SingleWMP

Template Step
K-Means
CCAEO

Construct Histogram Step

Ridge
Decision Tree
Random Forest
XGBoost
DNN

\( \mathcal{H}_1 \)

\( s \)

\( \hat{y} \)

\( \hat{y}_1, \ldots, \hat{y}_j, \ldots, \hat{y}_k \)\n
\( \sum_{j=1}^{k} \hat{y}_j = \hat{y} \)

\( \overline{y}_1, \ldots, \overline{y}_j, \ldots, \overline{y}_k \)\n
\( \sum_{j=1}^{k} \overline{y}_j = \hat{y} \)
Q2 - LearnedWMP Experimental Setup

LearnedWMP
- LearnedWMP-DNN
- LearnedWMP-Ridge
- LearnedWMP-DT
- LearnedWMP-RF
- LearnedWMP-XGB

SingleWMP
- SingleWMP-DNN
- SingleWMP-Ridge
- SingleWMP-DT
- SingleWMP-RF
- SingleWMP-XGB
- SingleWMP-DBMS
Q2 - LearnedWMP Accuracy Performance
Q2 - LearnedWMP Training Runtime Performance

Observation

<table>
<thead>
<tr>
<th>Model</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SingleWMP-DNN</td>
<td>46244.8</td>
</tr>
<tr>
<td>SingleWMP-Ridge</td>
<td>1912.7</td>
</tr>
<tr>
<td>SingleWMP-XGB</td>
<td>1867.9</td>
</tr>
<tr>
<td>SingleWMP-RF</td>
<td>127.9</td>
</tr>
<tr>
<td>SingleWMP-DT</td>
<td>1906.6</td>
</tr>
<tr>
<td>LearnedWMP-DNN</td>
<td>404.0</td>
</tr>
<tr>
<td>LearnedWMP-Ridge</td>
<td>96.1</td>
</tr>
<tr>
<td>LearnedWMP-XGB</td>
<td>9.7</td>
</tr>
<tr>
<td>LearnedWMP-RF</td>
<td>23.0</td>
</tr>
<tr>
<td>LearnedWMP-DT</td>
<td>9.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SingleWMP-DNN</td>
<td>1974.0</td>
</tr>
<tr>
<td>SingleWMP-Ridge</td>
<td>1.2</td>
</tr>
<tr>
<td>SingleWMP-XGB</td>
<td>38.0</td>
</tr>
<tr>
<td>SingleWMP-RF</td>
<td>11.0</td>
</tr>
<tr>
<td>SingleWMP-DT</td>
<td>58.6</td>
</tr>
<tr>
<td>LearnedWMP-DNN</td>
<td>15.8</td>
</tr>
<tr>
<td>LearnedWMP-Ridge</td>
<td>15.8</td>
</tr>
<tr>
<td>LearnedWMP-XGB</td>
<td>112.0</td>
</tr>
<tr>
<td>LearnedWMP-RF</td>
<td>9.9</td>
</tr>
<tr>
<td>LearnedWMP-DT</td>
<td>9.9</td>
</tr>
</tbody>
</table>

Motivation | Problem Definition | Methodology | Experimental Evaluation | Contribution & Future Work

Nicolas Andres Jaramillo Duran
Q2 - LearnedWMP Inference Runtime Performance

**TPC-DS**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (µs)</td>
<td>870.5</td>
<td>486.9</td>
<td>810.5</td>
<td>620.1</td>
<td>87.3</td>
<td>2467.3</td>
<td>77.0</td>
<td>110.5</td>
<td>58.0</td>
<td>10x faster</td>
</tr>
</tbody>
</table>

**TPC-C**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (µs)</td>
<td>942.9</td>
<td>572.1</td>
<td>1115.6</td>
<td>961.7</td>
<td>113.3</td>
<td>9695.8</td>
<td>78.6</td>
<td>313.5</td>
<td>1661.6</td>
<td>8x faster</td>
</tr>
</tbody>
</table>

**JOB**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (µs)</td>
<td>955.3</td>
<td>734.8</td>
<td>9322.6</td>
<td>874.6</td>
<td>98.1</td>
<td>63353.1</td>
<td>98.4</td>
<td>1334.5</td>
<td>108.4</td>
<td>9.7x faster</td>
</tr>
</tbody>
</table>

**Observation**

- SingleWMP-DNN TPC-DS: 870 µs, 10x faster
- LearnedWMP-DNN TPC-DS: 87 µs
- SingleWMP-DNN JOB: 955.3 µs, 9.7x faster
- LearnedWMP-DNN JOB: 98.1 µs
- SingleWMP-DNN TPC-C: 942.9 µs, 8x faster
- LearnedWMP-DNN TPC-C: 113.3 µs
Q2 - LearnedWMP Model Size

**Motivation**

**Problem Definition**

**Methodology**

**Experimental Evaluation**

**Contribution & Future Work**
Q3 - LearnedWMP Batch Size Parameter s

![Bar chart showing the relationship between MAPE (%) and Batch Size. The x-axis represents Batch Size ranging from 2 to 50, and the y-axis represents MAPE (%) ranging from 0 to 10. The chart displays bars for each batch size, with the highest MAPE at 10.4 for a batch size of 2, and decreasing values for higher batch sizes.](chart_image)
Contribution and Future Work
Contributions

- Introduce a novel problem of **workload memory prediction**

- Workload memory prediction as a **distribution regression problem**

- Propose **LearnedWMP**, a novel prediction **model** that can estimate the **memory** demand of a **batch of SQL queries**
  
  - Better Performance
  - Faster training and inference time
  - Smaller model size

- Our model reduced the memory estimation errors of **DBMS** by at least **47.16%**
Future Work - Datasets

» Current Datasets
  • TPC-DS
  • TPC-C
  • JOB

» Future Datasets
  • Production DB and dataset
Future Work – Resource Prediction

- Current Predicted Resource
  - Memory

- Future Resources to predict
  - Disk I/O
  - Throughput
  - CPU
Future Work – Multiple Resource Prediction

Single Resource Prediction

Multiple Resource Prediction

Motivation
Problem Definition
Methodology
Experimental Evaluation
Contribution & Future Work
Future Work – Learn Templates

Keep the structure of the query plan

RETURN
  CTQ
     48873.6
  HSJOIN
     48873.6

SORT
  134170

TBSCAN
  134170

HSJOIN
  1029

SPLIT
  3201

TBSCAN
  1603

TBSCAN
  770

TBSCAN
  3759.48

Encode queries
Thank you
Nicolas Andres Jaramillo Duran
Data Driven Industries

Motivation
Problem Definition
Methodology
Experimental Evaluation
Contribution & Future Work
Problem Definition

Definitions:

<table>
<thead>
<tr>
<th>Query $q = (e, p, m)$</th>
<th>$e$ is the query expression, $p$ is the query plan, and $m$ is the memory usage.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w = (Q, y)$</td>
<td>$Q$ is a set of queries where $q_i \in Q$ is a tuple $(e_i, p_i, m_i)$, and $y$ is the actual memory utilization of all queries in $Q$ such that $y = \sum_{i=1}^{Q} m_i$</td>
</tr>
<tr>
<td>$n$</td>
<td>number of workloads in the training corpus of the form ${(w_1, y_1), \ldots, (w_n, y_n)}$</td>
</tr>
<tr>
<td>$T = {t_1, \ldots, t_k}$</td>
<td>query template $t_i \in T$ represents a class of queries with similar memory requirements</td>
</tr>
<tr>
<td>$\mathcal{H} = [c_1, \ldots, c_k]$</td>
<td>let $w = (Q, y)$ be a workload, $c_i$ is the number of queries in $Q$ that can be mapped to query template $t_i \in T$. The counts of queries in $Q$ that map to different query templates in $T$ are recorded in a $1 \times d$ vector of length $k =</td>
</tr>
</tbody>
</table>

Assumptions:

- The distribution of queries among the query templates (i.e., the workload histogram bins) is uniform.
- The query templates are independently and identically distributed.
- An underlying function, $f(\cdot)$, exists that can accurately compute any workload’s memory usage, $y$, from the workload histogram, $\mathcal{H}$.

\[ f(\mathcal{H}) = y \]

We don’t know $f(\cdot)$ nor have access to the set of all possible workload examples to derive $f(\cdot)$. 
Problem Definition

Problem:
• We formulate estimating memory usage of an unseen workload as a distribution regression problem, where the estimate is computed from an input probability distribution - the distribution of queries $Q$ among templates $T$.
• For $w_i$, $H_i$ is the workload histogram and $y_i$ is the collective historical memory utilization of all queries in the workload.
• Let us assume we have a training corpus $\{(H_1, y_1), \ldots, (H_n, y_n)\}$ of $n$ workload histograms, one for each workload.
• Using distribution regression, we wish to learn a function, $\hat{f}(\cdot)$, an approximation of $f(\cdot)$.
• $\hat{f}(\cdot)$, can compute $\hat{y}$, an accurate estimate of the actual memory usage $y$

$$f(H) = \hat{y}$$
LearnedWMP Accuracy Performance

TPC-DS

TPC-C

Motivation  Problem Definition  Methodology  Experimental Evaluation  Contribution & Future Work
Phase 1: Learning Query Templates

- Learning rule-based query templates
  - Map the generated queries back to their corresponding templates.
  - Group queries based on their estimated cardinality similarity.

- Learning clustering-based templates
  - Query text feature encoding
    - Bag of words (BoW)
    - Word embedding encoding
    - Text mining approach
  - Query plan base feature encoding
    - Cardinality cost aggregation for each operator (CCAEO)
    - Query plan encoding while maintaining tree structure (QPEWMTS)
Phase 1: Learning Query Templates

- **Motivation**
- **Problem Definition**
- **Methodology**
- **Experimental Evaluation**
- **Contribution & Future Work**