Lecture #04

Carnegie Mellon University

*Special Topics:*

# Self-Driving Database Management Systems

## Index Recommendation II

@Poojan_Palwai // 15-799 // Spring 2022

# TODAY’S AGENDA

Background on Automated Indexing
Architecture of Automated Indexing
Testing of Automated Indexing
Reflection



Source: Lin Ma

# Automated Indexing

**Idea**

- Create/Use a set of indexes to reduce execution costs of queries:

**Goals**

- Ensure that creating and dropping indexes don't result in a query performance regression
- Make sure that query plans approached by the automatic indexing are in line with the optimizer

# Automated Indexing

## Insights

- **Classification** between two queries is better than regression since the indexer cares about the better plan

- For an **"in-sync"** with an optimizer, the only requirement is the indexes needs to **utilize the same plan**

# Algorithm’s Architecture

1. Convert query plans into fixed size feature vectors

1. Construct pairs of final feature vectors and obtain corresponding labels

1. Train an offline classifier with these features from aggregated databases

1. Localize the offline model to specific databases

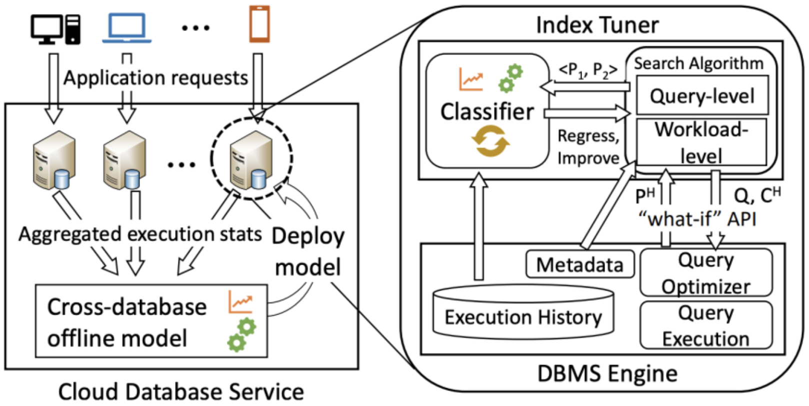



**Figure 2: Overview of an architecture leveraging the classifier trained on aggregated execution data from multiple databases in a cloud database service.**



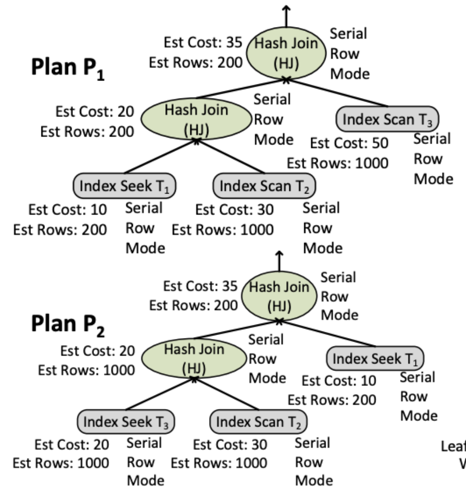
AI Meets AI: Leveraging Query Executions to Improve Index Recommendations
*Sigmoid 2019*

# Architecture: Generate Unique Vector

**Goal:** Vector has to be schema agnostic and fixed-size

**Solution**:

- Feature Vectors are # of unique operators, the parallelism of the operators, and execution mode by row or batch:
  - ⟨Physical Operator⟩_⟨Execution Mode⟩_⟨Parallelism⟩
- If multiple operators have the same key, sum up all the values assigned by the key
- The value of a node is determined by the sum of weight multiplied by the height of all its children




EstNodeCost (P1)

| | |
|---|---|
| Seek_Row_Serial | 10 |
| Scan_Row_Serial | 80 |
| HJ_Row_Serial | 55 |
| NLJ_Row_Serial | 0 |
| MJ_Row_Serial | 0 |
| … | … |

EstNodeCost (P2)

| | |
|---|---|
| Seek_Row_Serial | 30 |
| Scan_Row_Serial | 30 |
| HJ_Row_Serial | 55 |
| NLJ_Row_Serial | 0 |
| MJ_Row_Serial | 0 |
| … | … |

EstNodeCost (P2 – P1)

| | |
|---|---|
| Seek_Row_Serial | 20 |
| Scan_Row_Serial | -50 |
| HJ_Row_Serial | 0 |
| NLJ_Row_Serial | 0 |
| MJ_Row_Serial | 0 |
| … | … |

LeafWeightEst RowsWeighted Sum (P1)

| | |
|---|---|
| Seek_Row_Serial | 200 |
| Scan_Row_Serial | 2000 |
| HJ_Row_Serial | 4600 |
| NLJ_Row_Serial | 0 |
| MJ_Row_Serial | 0 |
| … | … |

LeafWeightEst RowsWeighted Sum (P2)

| | |
|---|---|
| Seek_Row_Serial | 1200 |
| Scan_Row_Serial | 1000 |
| HJ_Row_Serial | 6200 |
| NLJ_Row_Serial | 0 |
| MJ_Row_Serial | 0 |
| … | … |

LeafWeightEstRows WeightedSum (P2 - P1)

| | |
|---|---|
| Seek_Row_Serial | 1000 |
| Scan_Row_Serial | -1000 |
| HJ_Row_Serial | 1600 |
| NLJ_Row_Serial | 0 |
| MJ_Row_Serial | 0 |
| … | … |

AI Meets AI: Leveraging Query Executions to Improve Index Recommendations
*Sigmoid 2019*

# Architecture: Construct Pairs

**Label:**

- Context: A pair is labeled as regression if:

(ExecCost(P2) − ExecCost(P1)) / ExecCost(P1) > α (where α = 0.2)

- The pairs were combined using the same math:
  - Pair Diff: P2 - P1
  - Pair Diff Ratio: Pair Diff / P1
  - Since values sometimes became two large or small, they were either:
    - Gradient Clipped to (10^4) even if divided by 0
    - Normalized by the sum of attributes

# Architecture: Classifier

**Linear Models:**

- **Logistic Regression (LR)**

**Trees Models:**

- **Random Forest (RF) (bagging ensemble)**
- **Gradient Boosting Trees (GBT) (boosting ensemble)**
- **Gradient-Boosted Decision Trees (DGBM)**

# Architecture: Localization

- The local data was split into two subsets
  - The first subset was used to train a local model that used a Random Forest

  - The second subset was used to train a meta model which:
    - Tried to determine whether to use the local model or offline model
    - Features:
      - The local model and offline model's predictions
      - Uncertainty scores from the local and offline model
      - Nearest neighbor of both model to determine distance of feature vector of query plans from old data used in the models

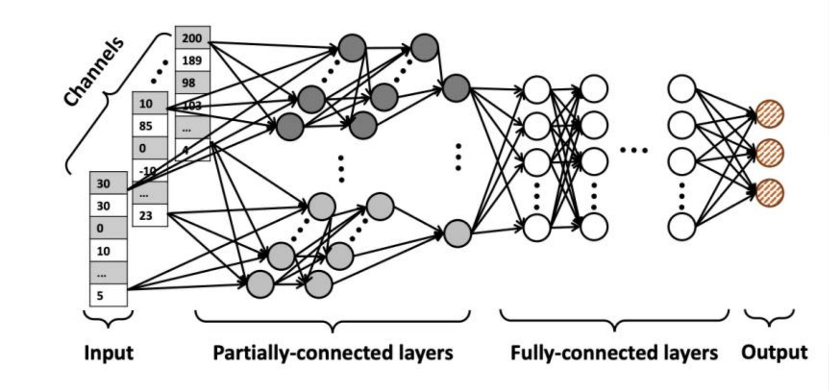
# Alternative Models

**Operator Level-Regression model: Proposed by Li et al. computed execution cost of each operators and combined them for the plan's execution cost**

**Plan-Level Regressor: Similar to Akdere et al. and predicted the execution cost of a plan**

**Deep-Neural Network:**

- **Partially-connected networks were used with similar operators**
- **Skip Connections that connected nodes from different layers**
- **Random-Forests: The network's last layer into a random forest**
- **Used Transfer Learning:**
  - **Initialize and freeze the weights of the DNN (offline)**
  - **Than train the model with new data by changing either the random forest or the final layer**




AI Meets AI: Leveraging Query Executions to Improve Index Recommendations
*Sigmoid 2019*

# Overall Testing

**Workloads:**

- Industry standard benchmarks: TPC-H (Skewed Data Generator) and TPC-DS
- Eleven workloads from customers:  SQL Server
- Two different scale factors: 10 and 100 that had same queries but different knobs

**Metrics:**

- Precision: Model's accuracy of positive prediction
- Recall: Model's coverage in correctly predicting the positives
- F1-Score: Harmonic mean of precision (P) and recall (R

**Data Splits for Train/Test:**

- Pairs:  Split the union of all plan pairs into disjoint sets
- Plans: Split the set of plans into two disjoint sets of plans from which the pairs are constructed.
- Query: Split the set of queries into two disjoint sets
- Database: Test Set is just a new database with unknown results

# Testing: Regression versus Classification

**Models:**

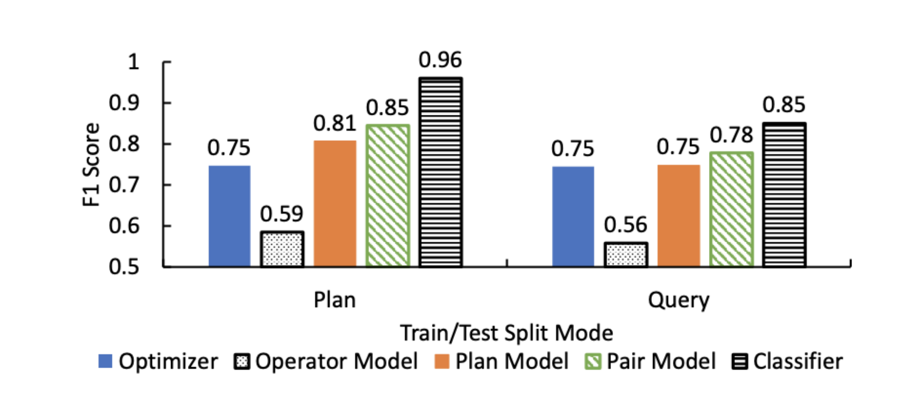
- Plan-Pair Model: GBT-Based Model with 250 Trees
- Plan_Level Model: RF-Based Model with 250 Trees
- Classifier: RF-Based Classifier

**Diff Ratio:**

- Cost difference in plans: max(cost1,cost2)/min(cost1,cost1) − 1)
- Plan-Model Used was Plan-Level Model

**Results:**

- 5× reduction in fraction of errors of Classifier over State-of-the-art Optimizer
- 2x Reduction in Errors of Classifier Over Plan Model




| Diff Ratio | 0.2 – 0.5 | | | 0.5 – 1 | | | 1 – 2 | | | > 2 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Plan Cost** | O | P | C | O | P | C | O | P | C | O | P | C |
| **0-25%** | | | | 0.70 | **0.84** | **0.84** | 0.74 | 0.92 | **0.93** | 0.85 | 0.96 | **0.97** |
| **25-50%** | 0.53 | 0.71 | **0.75** | 0.63 | 0.87 | **0.89** | 0.73 | 0.92 | **0.94** | 0.92 | 0.97 | **0.99** |
| **50-75%** | 0.53 | 0.77 | **0.84** | 0.62 | 0.90 | **0.93** | 0.71 | 0.95 | **0.97** | 0.92 | 0.98 | **0.99** |
| **75-100%** | 0.50 | 0.70 | **0.81** | 0.57 | 0.86 | **0.89** | 0.67 | 0.93 | **0.94** | 0.92 | 0.96 | **0.99** |

AI Meets AI: Leveraging Query Executions to Improve Index Recommendations
*Sigmoid 2019*

# Testing: Offline Model

**Models:**

- LR: Logistic Regression
- RF: Random Forest
- LGBM: Light Gradient-Boosted Decision Trees
- DNN: Deep Neural Network Without Random Forest
- Hybrid DNN: Deep Neural With Random Forest

**Training Time:**

- Random Forest trains in Tens of Minutes and infers Less than 10 Microseconds Per Data Point
- Deep Neural Network Trains in Couple of hours and Infer in 10s of Microsecond Per Data Point

**Results:**

- The Random Forest based models outperform others in accuracy and training efficiency

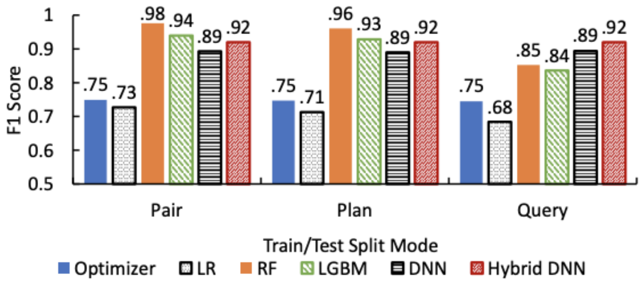



**Figure 7: Comparison of different modeling techniques for the classification task.**



AI Meets AI: Leveraging Query Executions to Improve Index Recommendations
*Sigmoid 2019*

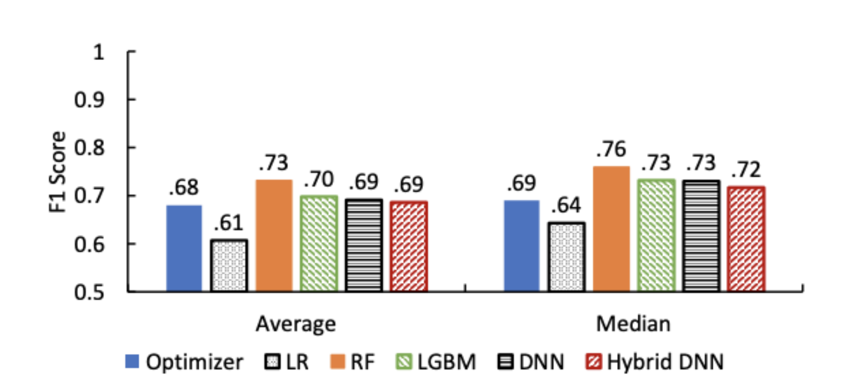
# Testing: Adaptation

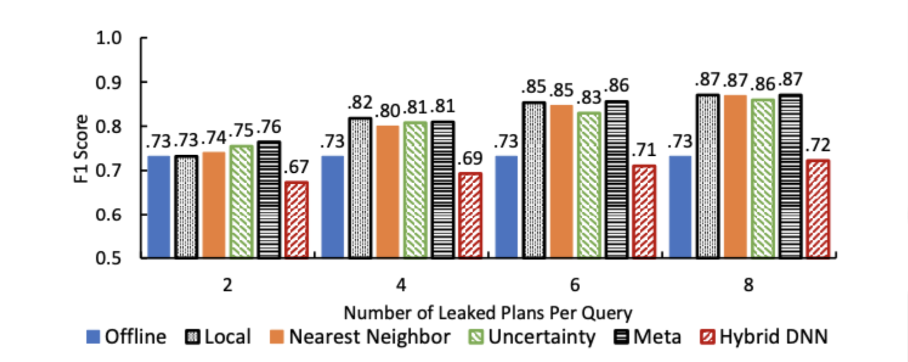
**Graphs:**

- The top graph displayed the offline models used with a database test-train split
- The bottom graph displayed the offline models incorporated with online learning techniques
  - All models (other than hybrid DNN) utilized random forests
  - Leaked plans represent the number of additional data the offline model had

**Results:**

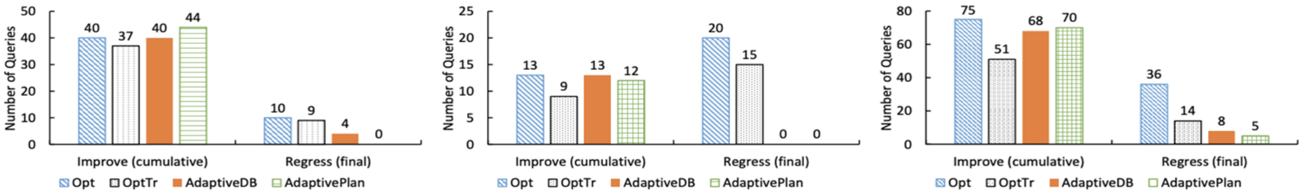
- The models with offline learning with random forest and meta-learning outperformed everything else including hybrid-DNN with transfer learning




AI Meets AI: Leveraging Query Executions to Improve Index Recommendations
*Sigmoid 2019*

# Testing: Index Recommendation




(a) TPC-DS 10g (b) TPC-DS 100g (c) Customer6

**Figure 11: Number of queries improved at its final configuration (with regressed configuration reverted) and regressed at the last iteration for query-level tuning with ten iterations.**

**Data:**

a. Improve (cumulative): # of queries Improved at least by 20% in final configuration
b. Regress (final): The number of queries that regress when the tuning stops.

Not Exclusive (It can improve and then regress)

**Workloads:** TPC-DS 10g with no index as initial configuration, TPC-DS 100g with existing columnstore as initial configuration, Customer 6 with no index as initial configuration

**Baselines:** Opt: Original index tuner with optimizer, OptTr: The index tuner with optimizer that uses a threshold to suggest plans



AI Meets AI: Leveraging Query Executions to Improve Index Recommendations
*Sigmoid 2019*

# PARTING THOUGHTS

**Variants:**

- Different search strategies
- Integrating index tuning with other physical design structures such as partitioning, materialized views, or column stores
- Formulating it as a continuous tuning problem
- Modeling robustness of physical design tuning

# References

[0] B.Ding et al., AI Meets AI: Leveraging Query Executions to Improve Index Recommendations, in *SIGMOD*, 2019