Database Native Approximate Query Processing Based on Machine-Learning

Yang Duan, Yong Zhang, Jiacheng Wu

University of Illinois Urbana-Champaign
Tsinghua University

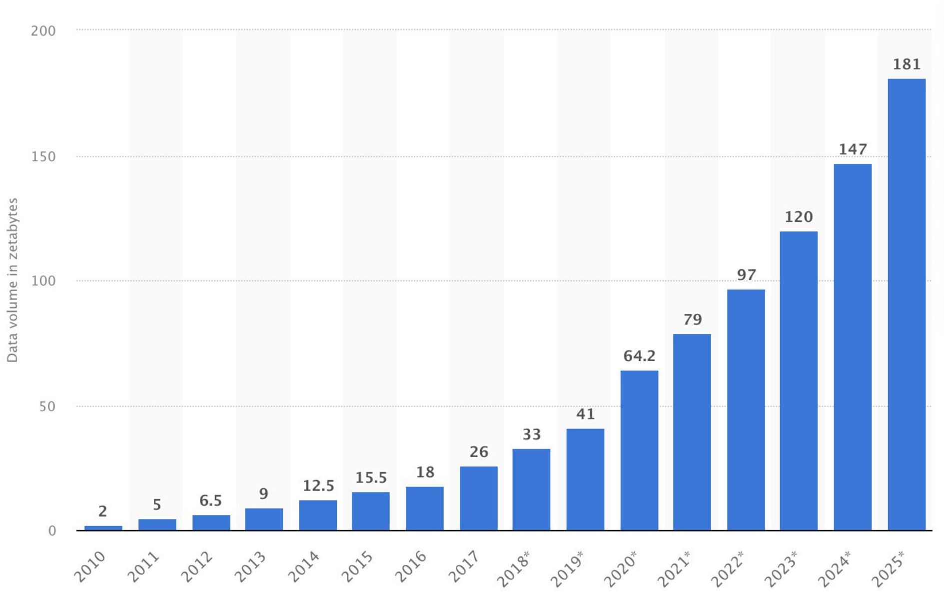
2021-09-25

Motivations - Database

1. It becomes an indispensable component in many industries

2. Volumes continuously expand with an incredible rate

3. Aggregation queries have unacceptable costs
   - e.g. Scan through billions of records to calculate the average expense that smartphone users spend per year



Trend of data volume from 2010 to 2025 [1]

[1] Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025. Statista.com

Motivations - Approximate Query Processing and Machine Learning

1. Approximate results are acceptable for aggregation queries in some case
   - e.g. 2000$ or 2005$ do not matter, several days -> several milliseconds
   - Latency, resource and accuracy need to be balanced

2. Machine Learning is designed for approximate computations / predictions.
   - Less space, less resource but better performance

Related Works and Objective

- Many researchers proposed well-designed AQP engines
  - BlinkDB [Eurosys, 2013]: a parallel, sampling-based approximate query engine
  - Taster [ICDE, 2019]: a self-tuning, elastic, online AQP engine

- Most researchers implemented their AQP engines separately
  - Usually, external servers are connected to host these engines

- Machine Learning has been studied in the area of AQP
  - regression with density-estimator models [SIGMOD, 2019]
  - Deep Learning to learn data distribution [ICDE, 2020]

- Based on the circumstance, we propose an idea of embedding the Machine Learning based AQP engine inside RDBMS

Methods - Neural Network

- Inclusion-exclusion Principle

$$
\begin{aligned}
&P[(b_{1L} < A_1 < b_{1U}) \ AND \ (b_{2L} < A_2 < b_{2U})] \\
&= (P[A_1 < b_{1U}] - P[A_1 < b_{1L}]) * (P[A_2 < b_{2U}] - P[A_2 < b_{2L}]) \\
&= P[A_1 < b_{1U} \ AND \ A_2 < b_{2U}] - P[A_1 < b_{1L} \ AND \ A_2 < b_{2U}] \\
&\quad - P[A_1 < b_{1U} \ AND \ A_2 < b_{2L}] + P[A_1 < b_{1L} \ AND \ A_2 < b_{2L}]
\end{aligned}
$$

Age	Height (cm)	Weight (kg)
5	115	18
5	117	17
15	165	49
16	171	51
21	176	55

Methods - Neural Network

- Convolutional Neural Network (CNN)
  - Better performance
  - Compatible
  - Favor mass data
  - Non-sequential data

Algorithm Decomposition and composition

Input: a list with bounds for corresponding columns in the order mentioned before

Output: Composed prediction r

1: Initialization: load trained model M

2: d $\leftarrow$ Decompose Input

3: s $\leftarrow$ Shuffle d

4: r $\leftarrow$ Obtain the prediction from M with s

Methods - User Defined Function (UDF)

“You can define simple functions that operate on a single row at a time, or aggregate functions that operate on groups of rows.” [1]

```
extern "C" double myAQP(UDF_INIT *initid, UDF_ARGS *args, char *is_null, char *error)

extern "C" my_bool myAQP_init(UDF_INIT *initid, UDF_ARGS *args, char *message)

extern "C" void myAQP_deinit(UDF_INIT *initid)
```

[1] Extending MySQL 8.0 / Adding Functions to MySQL / Adding a Loadable Function

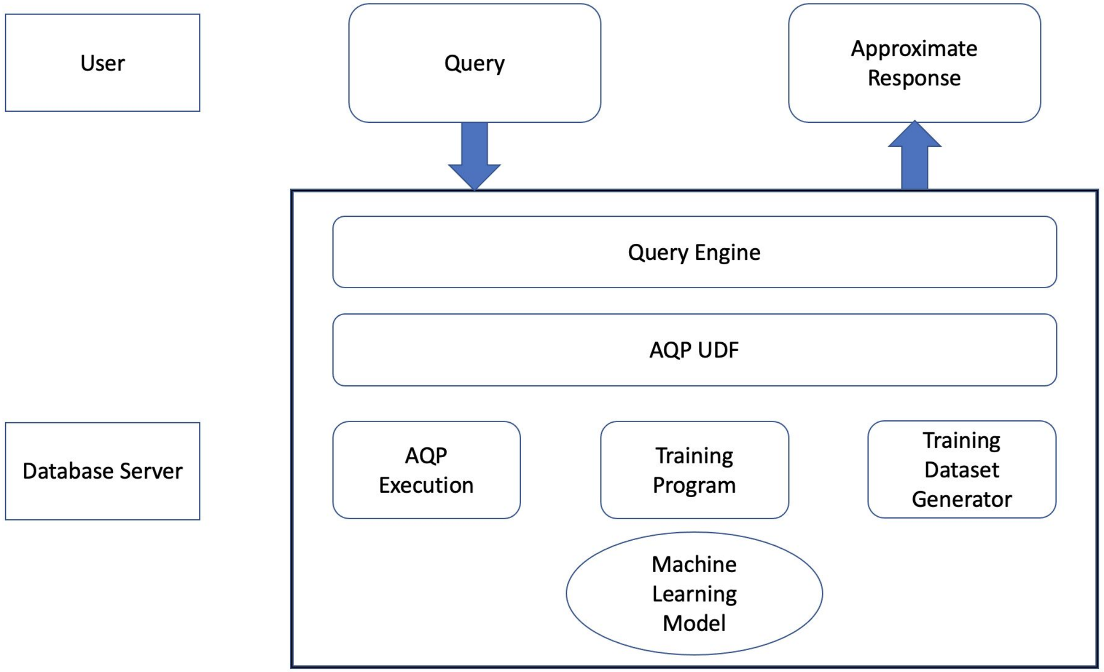
Methods - User Defined Function (UDF)

- Simple functions are ideal

- Training program is too resource-consuming and time-consuming
  - NVIDIA GeForce GTX1070
  - 2400 MHz 32GB memory
  - 10,000 training data points
  - Several hours

Methods - User Defined Function (UDF)

- External Training and External Query (ETEQ)
- External Training and Internal Query (ETIQ)



Methods - User Defined Function (UDF)

- External Training and External Query (ETEQ)

- External Training and Internal Query (ETIQ)

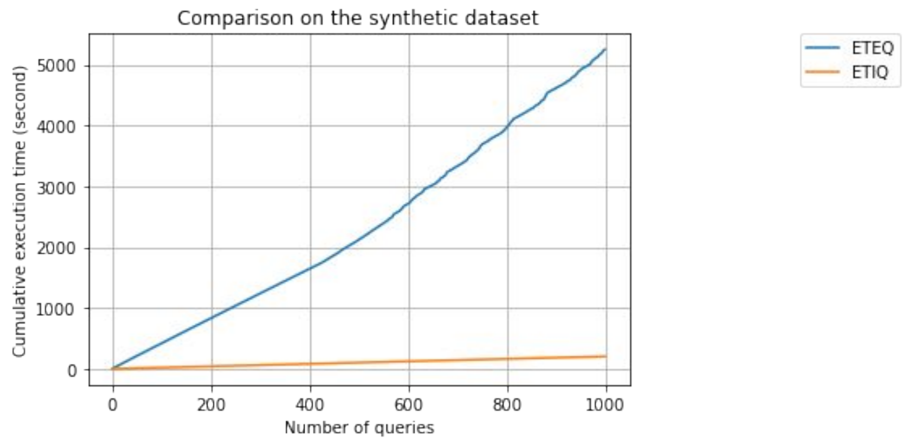
Implementation

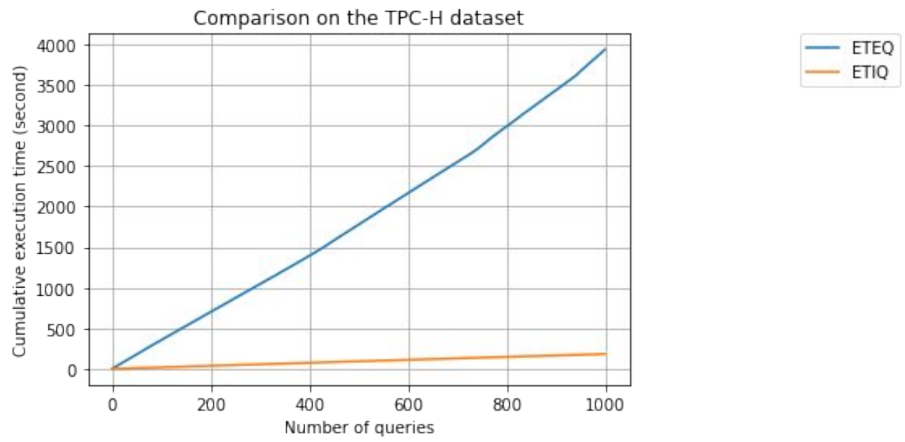
- Hardware
  - Intel i7-6820HK
  - NVIDIA GeForce GTX1070
  - 2400 MHz 32GB memory
- System
  - Windows Subsystem for Linux 2 (WSL2) with Ubuntu 18.04.
- Programming Language
  - Python 3.8.5
  - C++14
- Package
  - MySQL 5.7.34 on Ubuntu 18.04
  - CMake 3.19.2
  - GCC 7.5.0
  - Pytorch Stable 1.8.1
  - Cudatoolkit 11.0 on WSL2
  - Cudnn 8.2.0
  - Boost 1.69.0
  - Numcpp 2.4.2

Evaluation

- Two datasets of 1 million data points
  - Synthetic dataset
    - time as random seed
    - uniform distribution over [0, 1)
    - numpy.random.rand
  - TPC-H dataset

- Statistics
  - ETIQ is 26 times faster than ETEQ in term of the response time on the synthetic dataset
  - ETIQ is 22 times faster than ETEQ in term of the response time on the TPC-H dataset





Trends of ETIQ and ETEO

Conclusion and Future Works

- Embed the Machine Learning based AQP engine inside RDBMS
  - Present two different implementations in MySQL
  - Avoid unnecessary work of setting up external servers
  - Enable users to extend functionalities of other tools of RDBMS

- Synchronization is the main focus on the future work
  - Better training algorithms to learn the updating distribution
  - Automatically retrain and update models

Thanks for Your Time!

Code: https://github.com/thu-west/Learned-AQP

Email: yangd4@illinois.edu

Github: https://github.com/duanyang25