**QO-Insight – Inspecting Steered Query Optimizers**

Christoph Anneser† Mario Petruzelli‡ Nesime Tatbul†‡ David Cohen∞
Zhenggang Xu∞ Prithviraj Pandian∞ Nikolay Laptev∞ Ryan Marcus∞ Alfons Kemper†
†Technical University of Munich ‡Intel †MIT ∞Meta ∞University of Pennsylvania

**Background – Steered Query Optimizers**

- Database systems expose knobs that can be used to steer query execution. For example, PostgreSQL has knobs to disable nested loop joins or index scans.
- **Hint-sets** can combine multiple knobs.
- Recent work on steered query optimizers either adaptively selects [1], predefines [6] or randomly chooses [8, 13] multiple hint-sets, which are used to generate alternative query plans.
- A deep neural network predicts the execution time of each plan.

**Data Generation §3.1**

- DBMS e.g., PostgreSQL
- [1, 6, 8, 13]
- Steering Tool e.g., [1, 6, 8, 13]
- Results serialize
- ETL
- Data Ingestion
- Rust Server §3.2
- Ingestion SQL/JSON
- User Client cf. §3.3 and Figure 2

**Related Work**

- [8] Negi et al.: “Query Optimizer: A Practical Take on Big Data Workloads” (SIGMOD’21)
- [13] Zhang et al.: “Deploying a Steered Query Optimizer in Production at Microsoft” (SIGMOD’22)

**Scenario I: Database Admins**

**Goal:** Tune the database for a custom workload or query

> Switch to **query-centric** mode in A
> Adapt the performance metric in C
> Sort the table in E by the performance score
> Select the best hint-set for the query and apply it

**Scenario II: Query Optimization Experts**

**Goal:** Improve the query optimizer’s implementation

> Switch to **rule-centric** mode in A
> Click on two bars in the bar chart and open F
> Compare the two query plans
> Improve the query optimizer implementation