Lecture #19

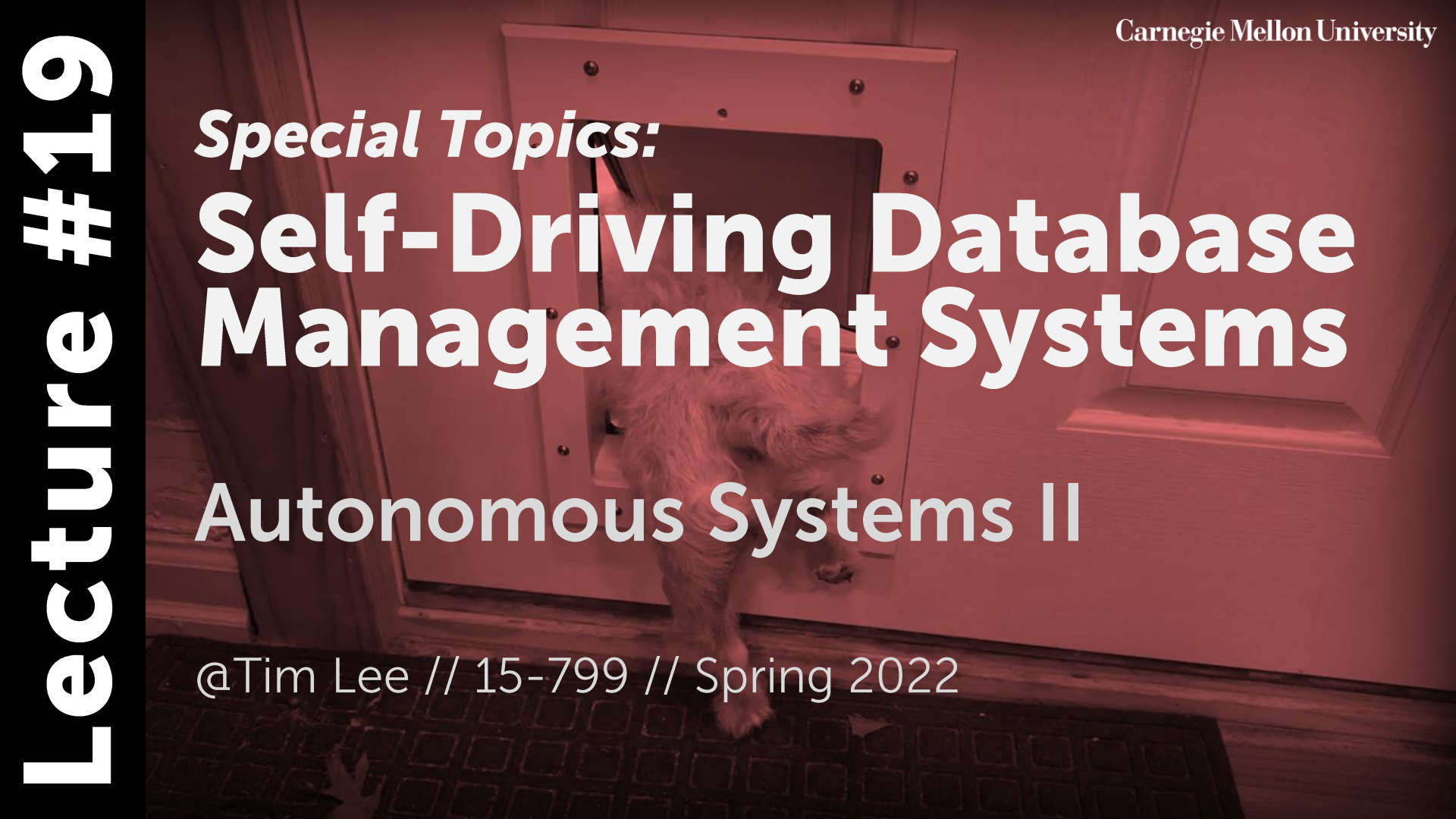
Carnegie Mellon University

Special Topics:

Self-Driving Database Management Systems

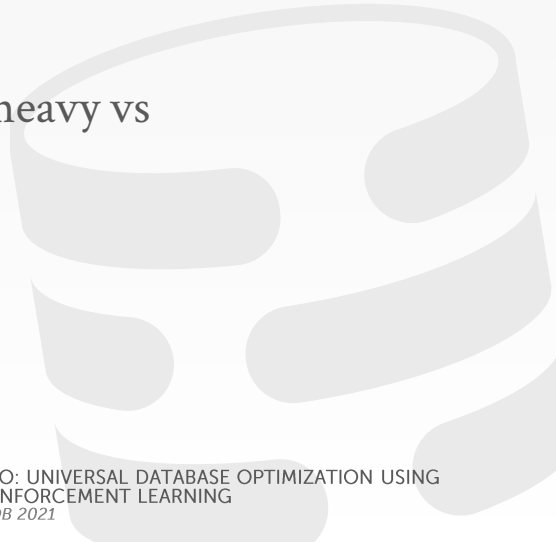
Autonomous Systems II

@Tim Lee // 15-799 // Spring 2022



LAST CLASS (UDO)

- RL-based database tuning tool that considers multiple types of tuning
- Actually running samples
  - high-quality & high-overhead
- Reduce reconfiguration overhead by distinguishing heavy vs light parameters



UDO: UNIVERSAL DATABASE OPTIMIZATION USING REINFORCEMENT LEARNING
VLDB 2021

TODAY'S AGENDA

- Overview of openGauss
- Learned Optimizer
- Learned Advisor
- Model Validator
- Experiment

TODAY'S AGENDA

- Overview of openGauss
- Learned Optimizer
- Learned Advisor
- Model Validator
- Experiment

openGauss

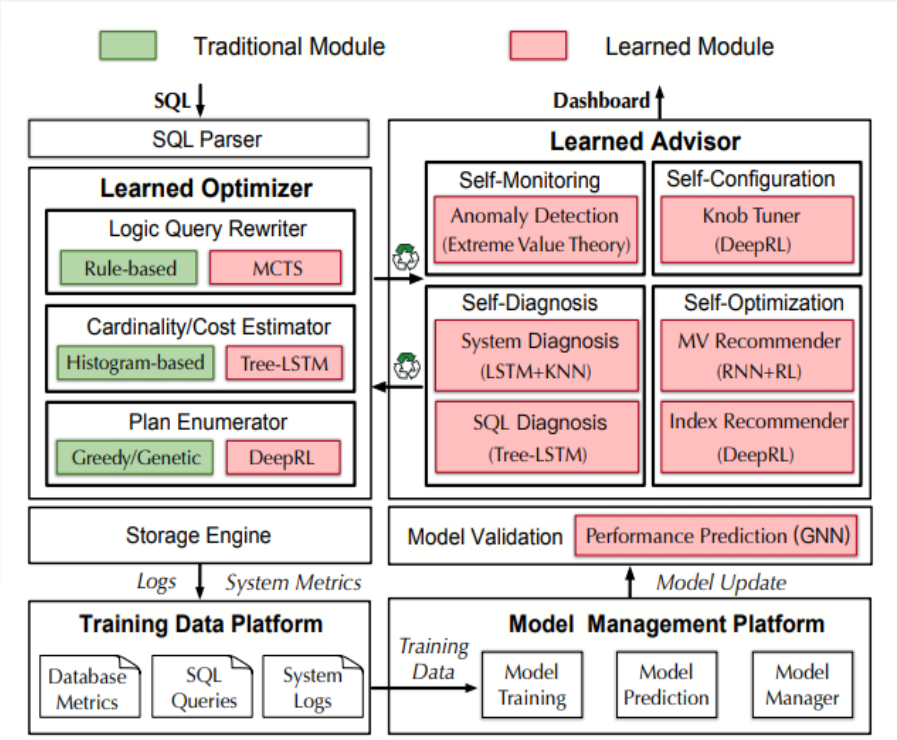
- Intro Video
- openGauss = Huawei GaussDB + open-source + Tsinghua University's autonomous DB research
  - GuassDB
  - Forked from PostgreSQL 9.4
  - multiprocess -> multithread
  - C -> C++

Challenges

1. Model Selection
   - DL models for cost estimation to capture correlation between columns
   - DRL models for knob tuning to tackle high-dimensional continuous space
2. Model Validation (= behavior modeling)
   - How to validate learned models are better than non-learning methods
3. Model Management
   - Unified ML platform for different models in different components
4. Training Data Management

openGauss

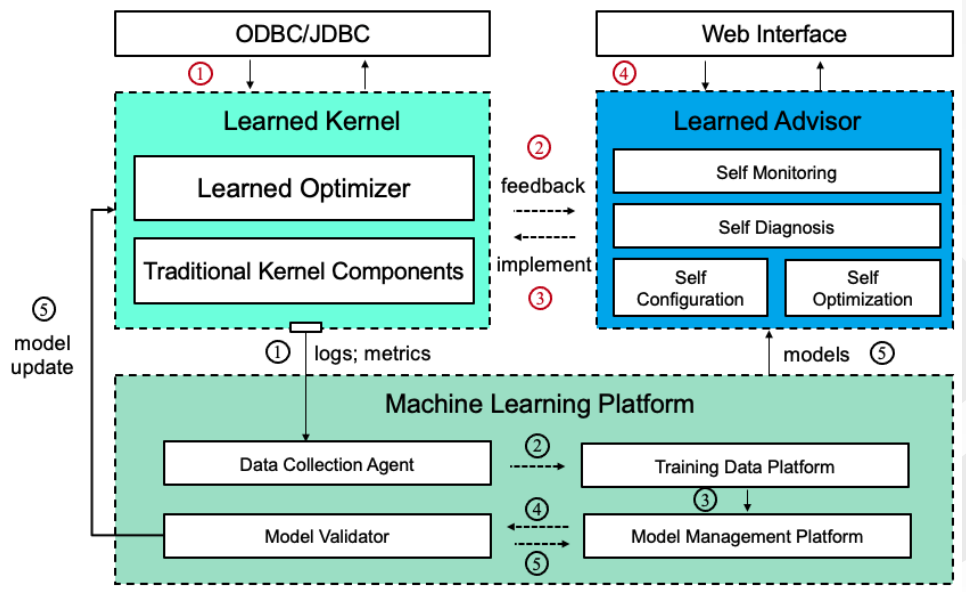
- Learned optimizer
  - query rewriter
  - cost/cardinality estimator
  - plan enumerator
- Learned advisor
  - Anomaly detection
  - knob tuning
  - materialize view / index selection
- Model Validator
  - Predict the performance before deploying a learned model
- Training Data Platform
- Model Management Platform



Data Flow / Implementation

1. Kernel optimizes submitted queries
2. Advisor monitors via RPC
3. If anomaly is detected, calls corresponding optimization function

1. ML platform: periodically collects data and fine-tune models
2. If a model is verified by the model validator, deploys the model



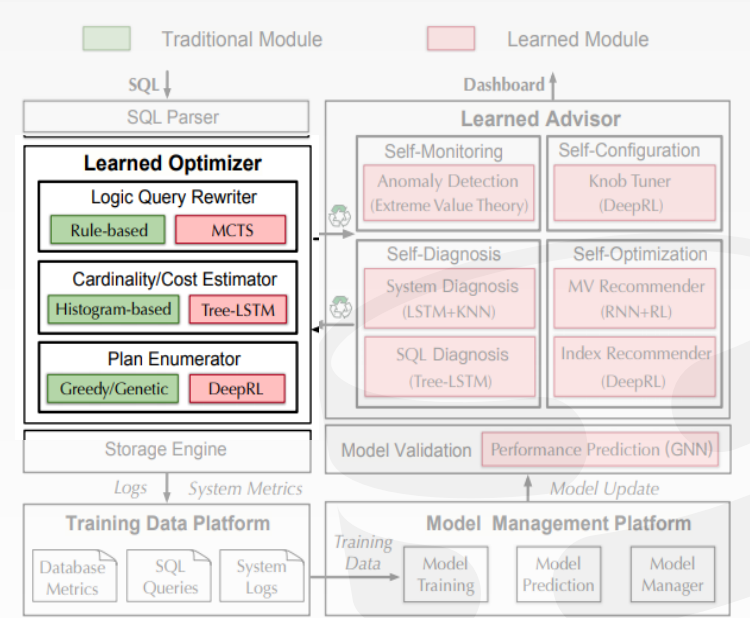
Source: Guoliang Li

openGauss – previous papers

- An Autonomous Materialized View Management System with Deep Reinforcement Learning – ICDE 2021
- QTune: A Query-Aware Database Tuning System with Deep Reinforcement Learning – VLDB 2019
- An End-to-End Learning-based Cost Estimator – VLDB 2019
- Reinforcement Learning with Tree-LSTM for Join Order Selection – ICDE 2020
- Query Performance Prediction for Concurrent Queries using Graph Embedding – VLDB 2020

TODAY'S AGENDA

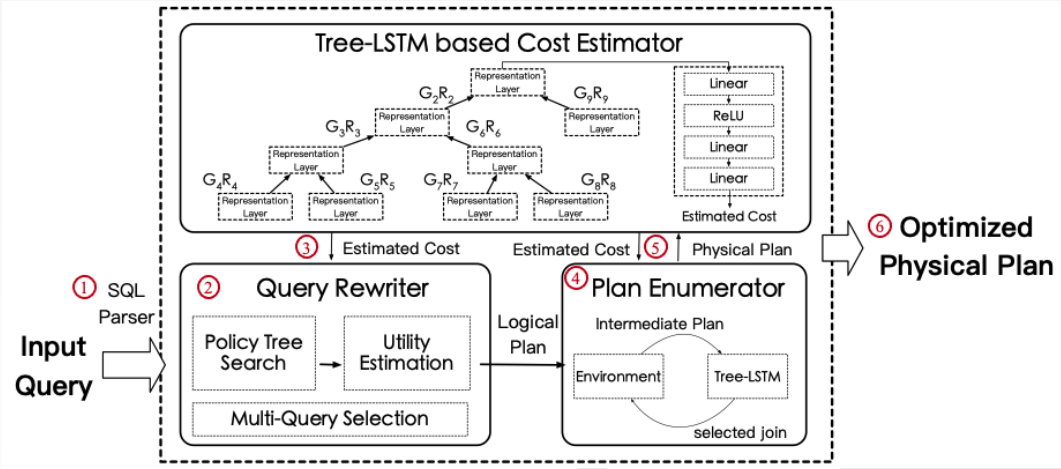
- Overview of openGauss
- Learned Optimizer
- Learned Advisor
- Model Validator
- Experiment



Learned Optimizer

For every incoming query:

- Query Rewriter
  - Output optimized logical plan
  - Calls cost estimator component to compute cost reduction (???)
- Cost Estimator
  - Estimate the plan execution cost
- Plan Generator
  - Optimize query in physical operator level



Source: Guoliang Li

Query Rewriter

- **Transform a slow SQL query into an equivalent one with higher performance**
- Monte Carlo Tree Search
  - MCTS finds rewrite orders that gain the most cost reduction
  - Relies on Cost Estimation model
  - Upper Confidence Bounds (UCB) to expand promising / uncovered branches

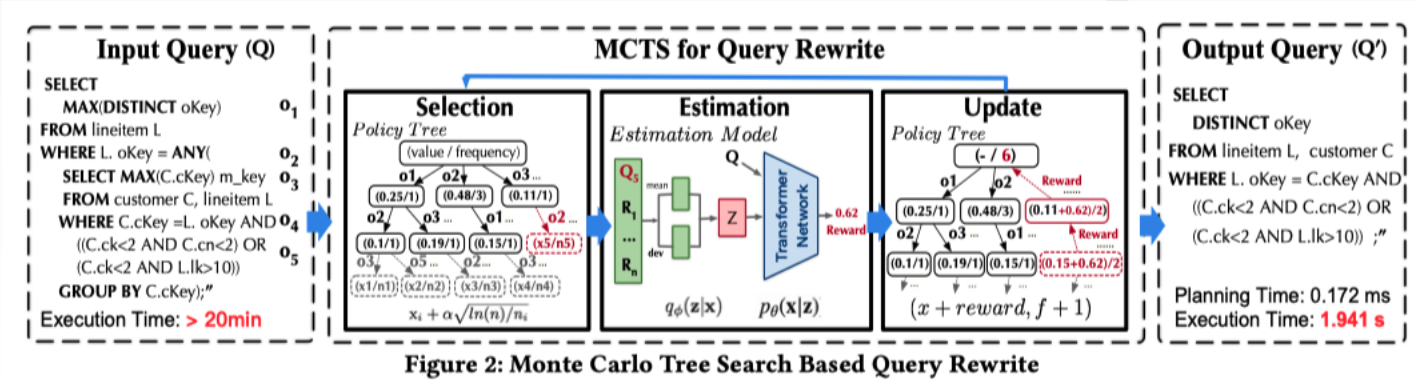


Figure 2: Monte Carlo Tree Search Based Query Rewrite

Learned Cost Estimator

- **Evaluates cost & cardinality of a query plan**
- Traditional cardinality estimation can't capture correlations between different columns
  - (make="Honda" AND model="Accord")
- Neural Networks to the rescue
  - Tree-structured LSTM
  - Embed both queries and physical plan
  - Extract representation of a plan tree recursively
  - Representation Memory Pool

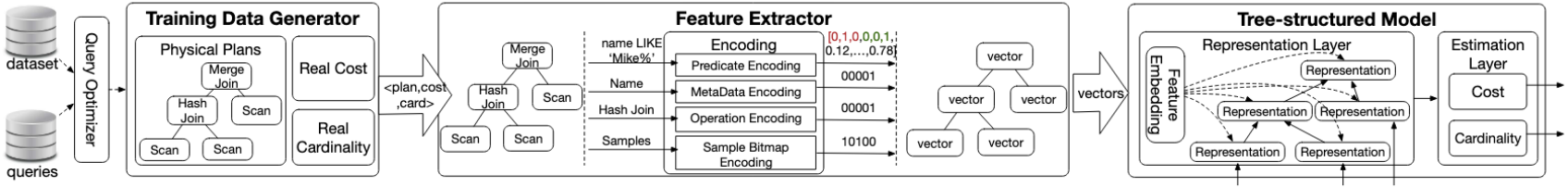
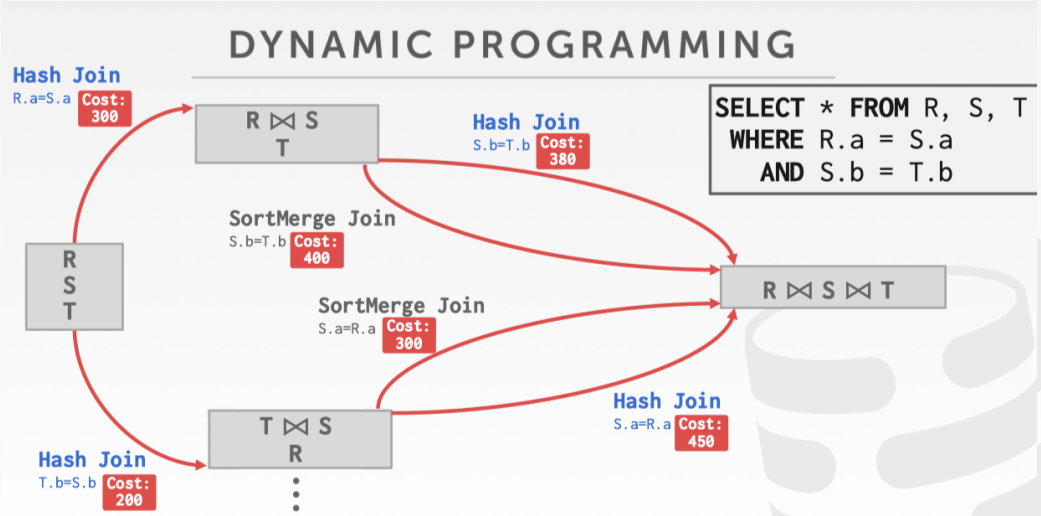


Figure 2: Architecture of learning-based cost estimator

Learned Plan Enumerator

- Deal with 2 tasks
  - **Join order selection**
  - **Physical operator selection**
- Dynamic programming
  - Search solution space based on cost model
  - Takes exponential time
- Deep Reinforcement Learning
  - Model plan generation as Markov decision process
  - Takes polynomial time
  - Can't handle schema update due to fixed-length representation of join tree



REINFORCEMENT LEARNING WITH TREE-LSTM FOR JOIN ORDER SELECTION
ICDE 2020

Learned Plan Enumerator

- **openGauss: Deep Q-Network with Tree-LSTM**
  - To capture tree structure of the plan tree
  - DQN uses Q-Network to find the next action
- An Action: (table1, table2, operator) is retrieved according to the q-value from the Tree-LSTM network
- RL keep choosing actions until whole plan tree is built
- Plan execution result is used to train Q-Network

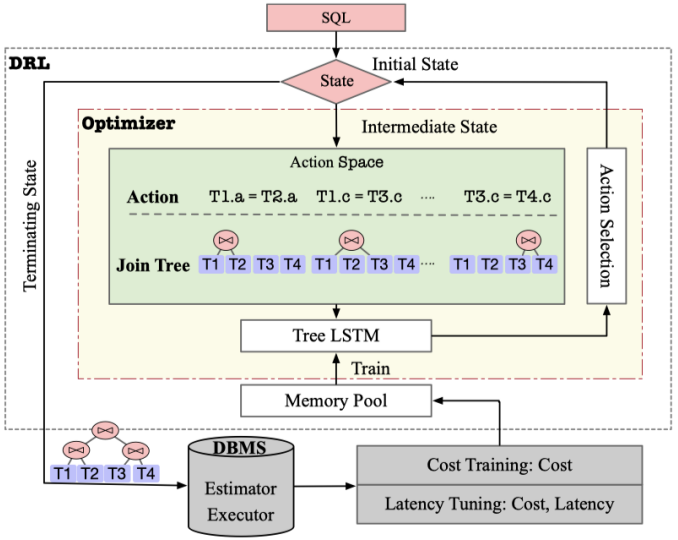
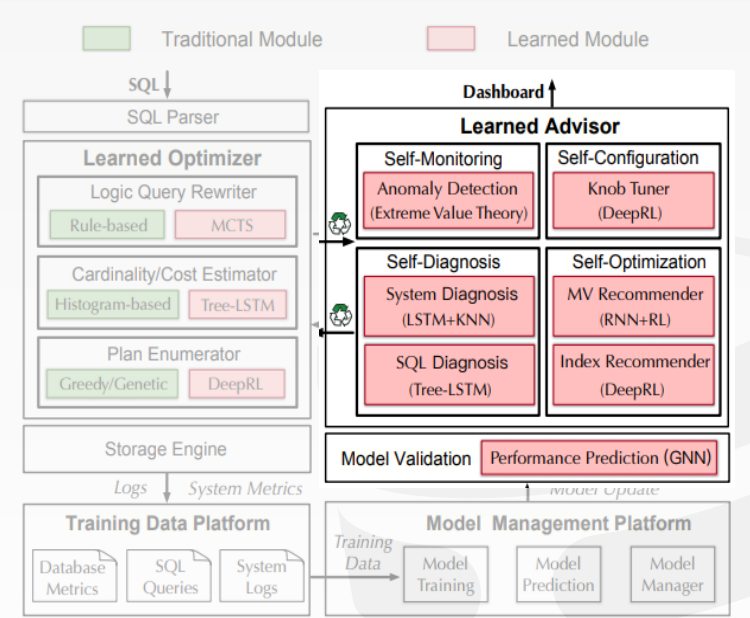


Figure 4: DRL Based Plan Generator

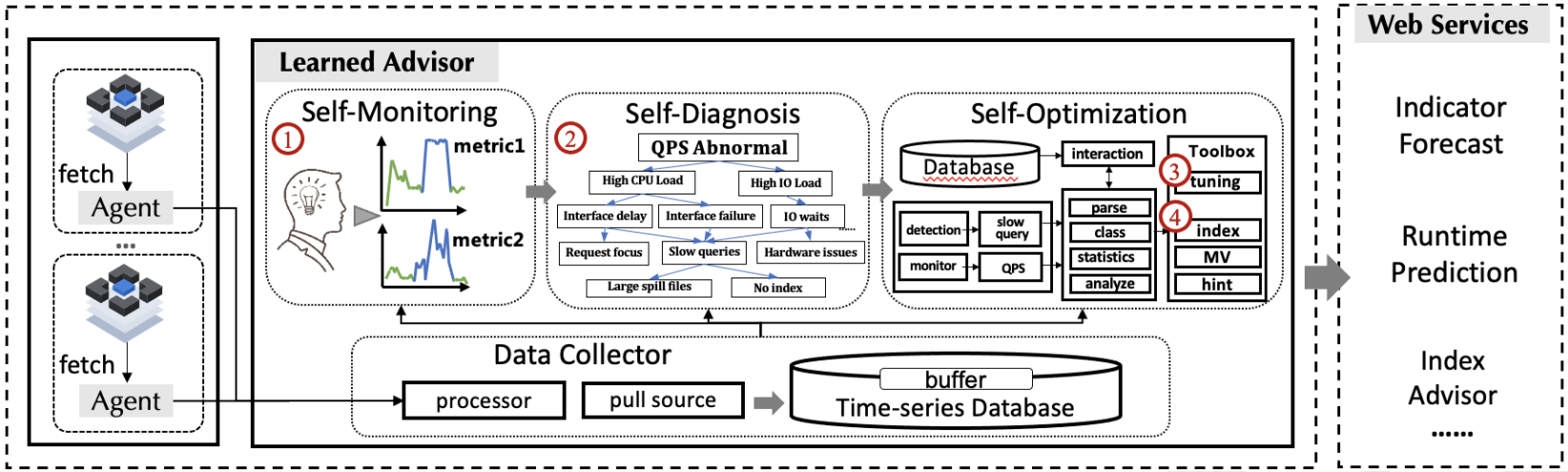
REINFORCEMENT LEARNING WITH TREE-LSTM FOR JOIN ORDER SELECTION
ICDE 2020

TODAY'S AGENDA

- Overview of openGauss
- Learned Optimizer
- Learned Advisor
- Model Validator
- Experiment



Learned Advisor



- **Self-Monitoring:** Collects runtime metrics (CPU / RT / sys logs); Detects anomalies
- **Self-Diagnosis:** Find out root causes of system anomalies
- **Self-Configuration:** Adopts DRL to tune knobs
- **Self-Optimization:** Recommends indexes & materialized views

Source: Guoliang Li

Self-Monitoring

- **Basic idea: when the metrics is far from the prediction result, there may be an anomaly**
- Monitors 500+ metrics
  - CPU/memory/disk usage; response time; cache hit
- LSTM-based auto-encoder with attention layer
  - Continuously train with time-series data
  - If reconstruct error > threshold => anomaly
  - **Extreme Value Theory** to calculate threshold dynamically

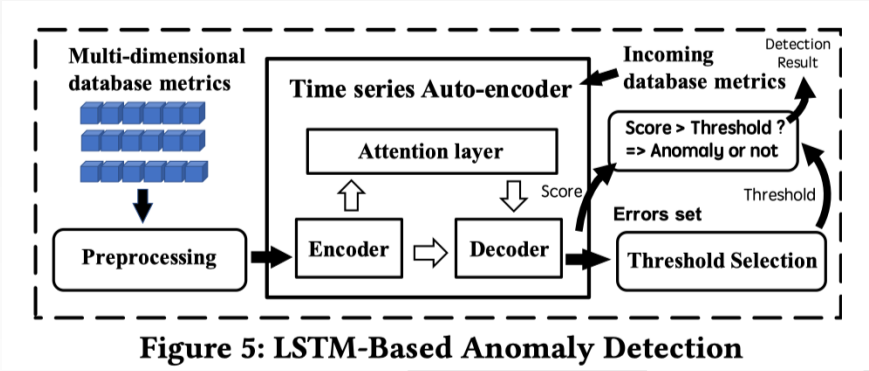


Figure 5: LSTM-Based Anomaly Detection

Self-Diagnosis

- **Diagnose root causes for anomalies**
- System Diagnosis
  - Classify metrics anomalies to system-level issue (IO contention, network congestion, insufficient disk space etc.)
- Offline training
  - Train LSTM based on normal data
  - Extract representative metrics in abnormal data
  - Labeling anomaly data requires human effort
- Online diagnosis
  - Kolmogorov-Smirnov test to generate anomaly vector
  - Finds similar anomaly cases

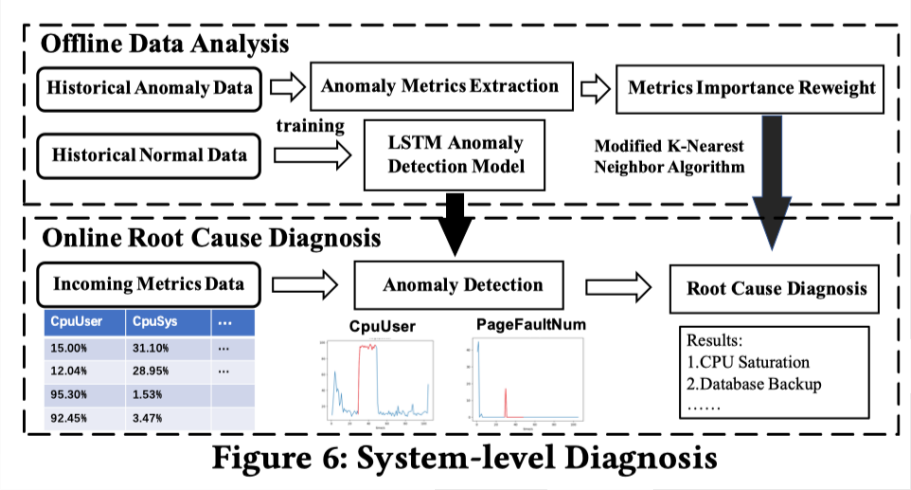


Figure 6: System-level Diagnosis

Self-Diagnosis

- SQL Diagnosis
  - Find out time-consuming operators (bottleneck) in a slow SQL query
- Model training
  - Input: A vector that encodes SQL query plan and **system metrics** (e.g. IO contention might make seq scan slower)
  - Output: slowest operator (labeled by running logs or **experts**)

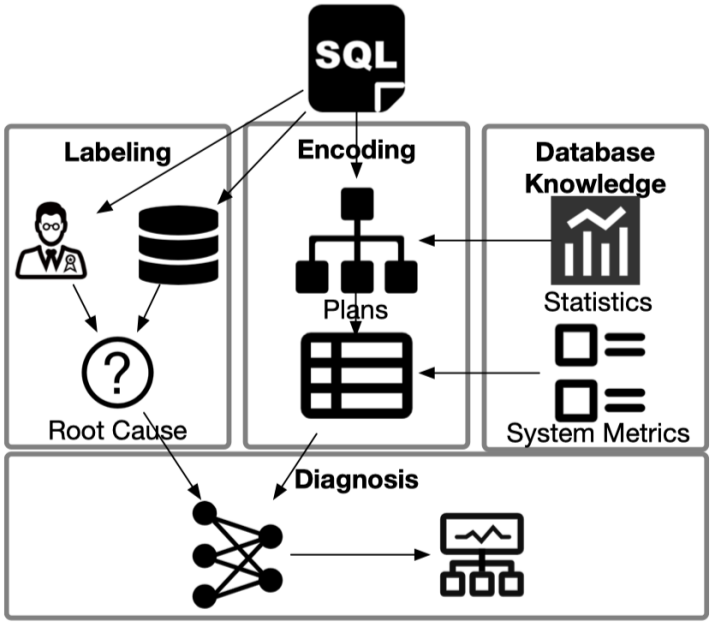


Figure 7: SQL-level Diagnosis

Self-Configuration

- **Tuning 400+ knobs in openGauss**
- DB Side
  - Extract internal metrics and current knob settings
- Algorithm Side
  - Search-based algo, DRL (Q-learning, DDPG)
- Tuning Side
  - Input DB status, query features
  - Output recommended knob values
- Workload Side
  - Run benchmark tests, provide execution performance as reward

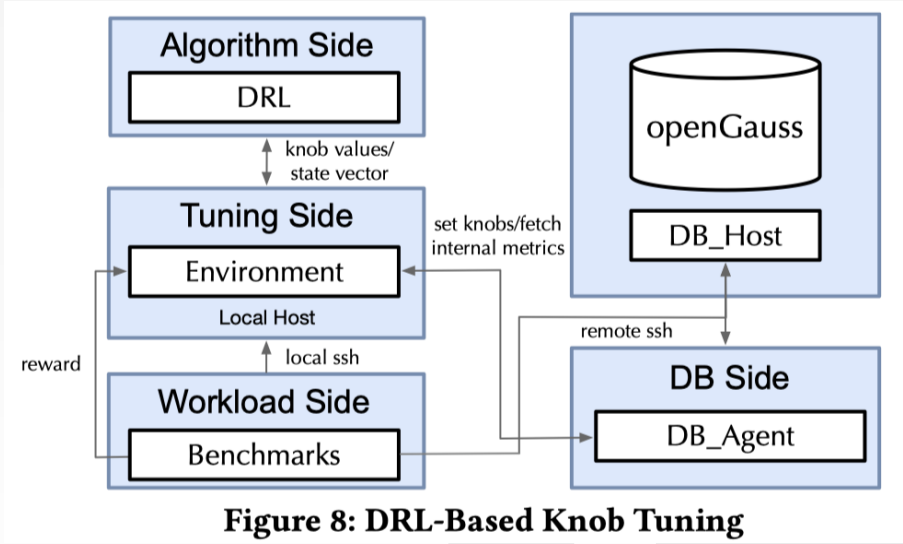


Figure 8: DRL-Based Knob Tuning

Self-Configuration

- **3 Modes**
- Rule Mode
  - Reduce tuning space
  - Generate knob settings based on rules
  - Report unreasonable knob values
- Training Mode
  - Tryout knob values, train RL model
- DRL Mode
  - Uses optimization algorithms for knob tuning
  - Global Search algo: doesn't need to be trained but worse performance
  - DRL: needs training but better performance

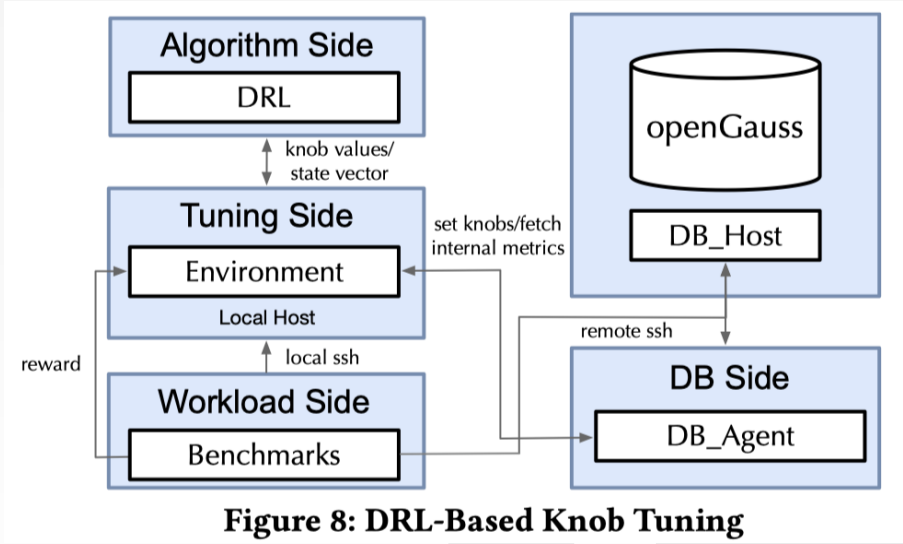


Figure 8: DRL-Based Knob Tuning

Self-Optimization

- **Materialized View Recommender**
- MV: caching the results of a query
  - A space-for-time trade-off
- How to choose candidate?
  - Select **subqueries** with high frequency and computation cost
- MV estimation
  - Predict the benefit (saved execution time), cost (space, generation time) for a set of MVs
  - Input data: queries & MV features
- MV selection
  - Maximize benefit given space budge using RL

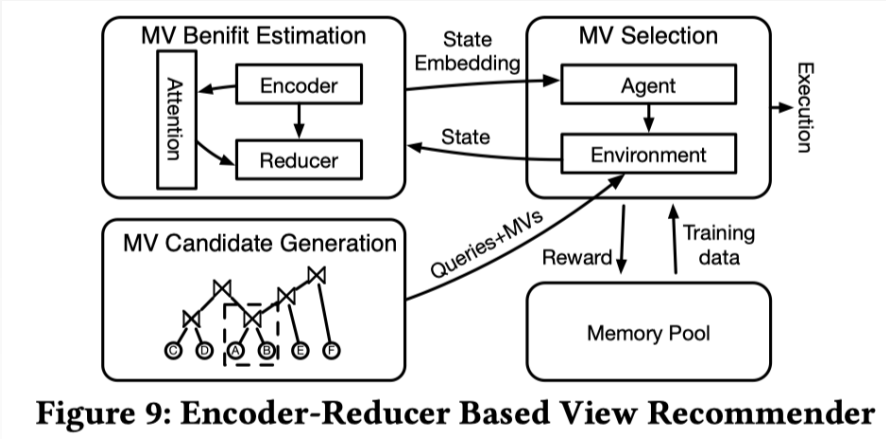


Figure 9: Encoder-Reducer Based View Recommender

Self-Optimization

- **Index Recommender**

Steps:

1. Extract representative queries in entire workload
2. Score columns based on access frequency and table statistics -> candidate indexes
3. Estimate benefit for candidate by **Hypo Index**
4. Deep Q-Network to pick indexes that maximize the benefit

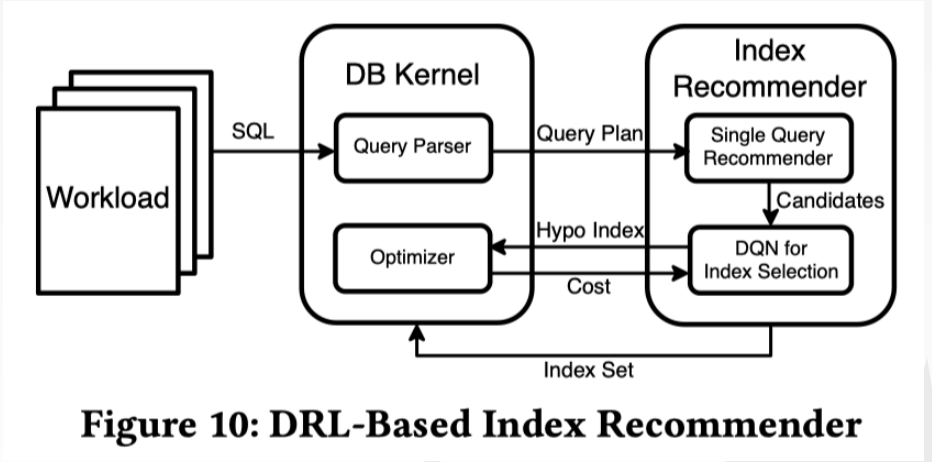
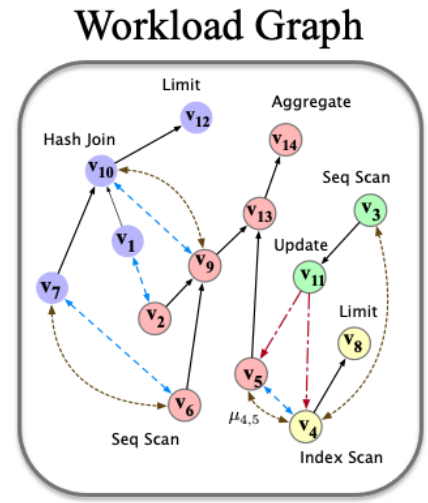


Figure 10: DRL-Based Index Recommender

TODAY'S AGENDA

- Overview of openGauss
- Learned Optimizer
- Learned Advisor
- Model Validator
- Experiments

Model Validation



- **Verify if a model is worth deploying**
  - If improves query performance, deploy it
  - Otherwise, drop it
- Embed workload as graph to capture correlation between operators
  - Parent-child relation
  - Read-write / write-write conflict
  - Data sharing
- **Optimization actions can be encoded into features in the Workload Graph**
- Feed graph in DL model to get performance
- Why do we still need Cost Estimator?

Actions	Graph Features
• Rewrites/Hints/JoinOrders	• Vertex Features
• Tuned Knob Values	• Configuration Relations
• Created Indexes	• Vertex Features; • Configuration Relations
• Created MVs	• Vertex Features; • Configuration Relations
…	…

TODAY'S AGENDA

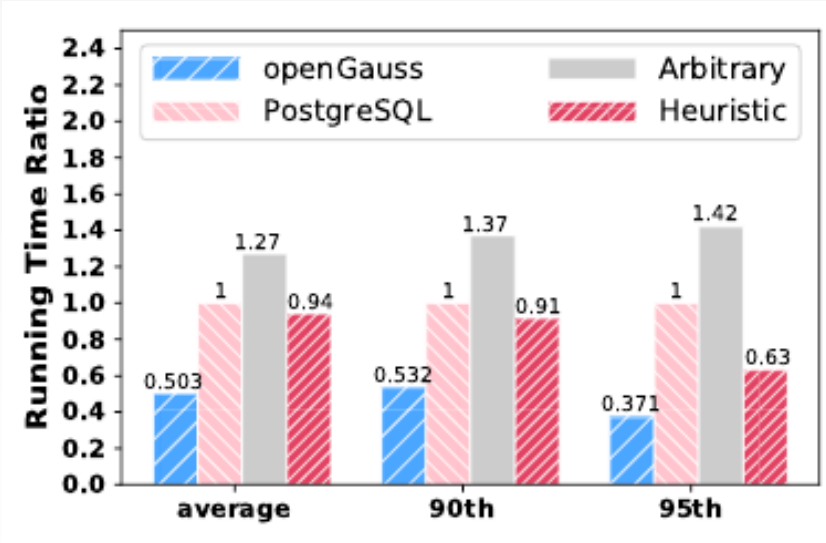
- Overview of openGauss
- Learned Optimizer
- Learned Advisor
- Model Validator
- Experiments

Datasets

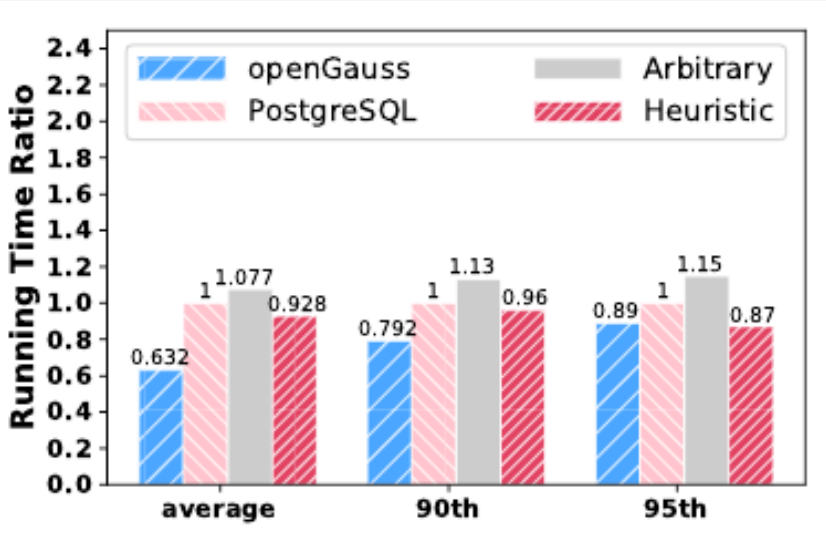
Name	Mode	Table	Size(GB)	#Query
JOB	RO	21	3.7	113
TPC-C	RW	9	1.30	912,176
TPC-H	RO	8	1.47	22
TPC-DS	RO	25	1.35	99

Learned Query Rewrite

- Able to explore different rewrite order
- TPC-H: perform well because lots of subqueries are removed
- JOB: lots of multi-joins, would be optimized by plan enumerator



TPC-H



JOB

Learned Cost Estimation

- TLSTM-Hash/Emb: different string embedding in predicate
- TPool: enable MIN-MAX pooling for complex predicate
- Why only JOB workload?
- Not comparing with other learned cost estimator

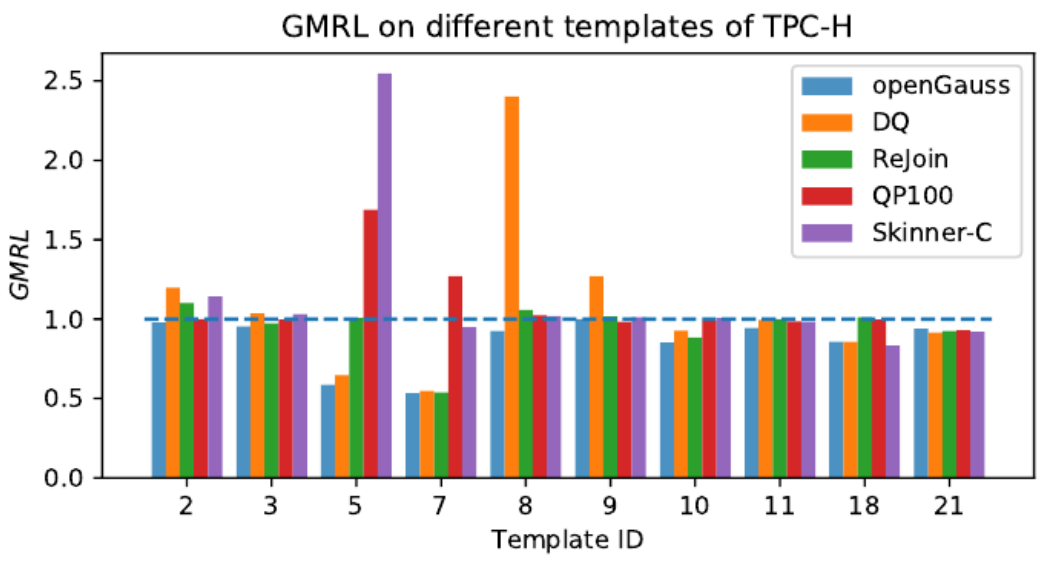
Table 2: Cost Estimation – Test errors on the JOB workload

Cardinality	median	90th	95th	99th	max	mean	Cost	median	90th	95th	99th	max	mean
PostgreSQL	184	8303	34204	106000	670000	10416	PostgreSQL	4.90	80.8	104	3577	4920	105
MySQL	104	28157	213471	1630689	2487611	60229	MySQL	7.94	691	1014	1568	1943	173
Oracle	119	55446	179106	697790	927648	34493	Oracle	6.63	149	246	630	1274	55.3
openGauss (TLSTM-Hash)	11.1	207	359	824	1371	83.3	TLSTM-Hash	4.47	53.6	149	239	478	24.1
openGauss (TLSTM-Emb)	11.6	181	339	777	1142	70.2	TLSTM-Emb	4.12	18.1	44.1	105	166	10.3
openGauss (TPool)	**10.1**	**74.7**	**193**	**679**	**798**	**47.5**	TPool	**4.07**	**11.6**	**17.5**	**63.1**	**67.3**	**7.06**

Learned Plan Enumerator

- Dynamic programming as baseline (GMRL = 1)
- Outperforms DQL methods (DQ, ReJoin), heuristic (QP), MCTS (Skineer-C)
- TPC-H: shorter, limited search space

	JOB	TPC-H
openGauss	**0.67**	**0.92**
ReJoin	1.14	0.96
QP100	NA	1.03
QP1000	1.90	1.00
Skinner-C	0.89	1.03
DQ	1.23	0.99



Learned Advisor

- Lack comparison with other ML techniques

Table 4: Knob Tuning Comparison – openGauss (R) denotes rule based tuning, openGauss (D) denotes DRL based tuning.

	TPC-H (s)	JOB (s)	TPC-C (tpmC)
PostgresQL	121.3	220.19	5552
DBA	95.1	193.37	7023
openGauss (R)	94.3	192.81	7574
openGauss (D)	**82.7**	**163.88**	**12118.4**

Table 5: Index Advisor

	TPC-H (s)	TPC-C (tpmC)
openGauss	**122.9**	**10202**
DBA	130.1	10001
Default	140.8	9700

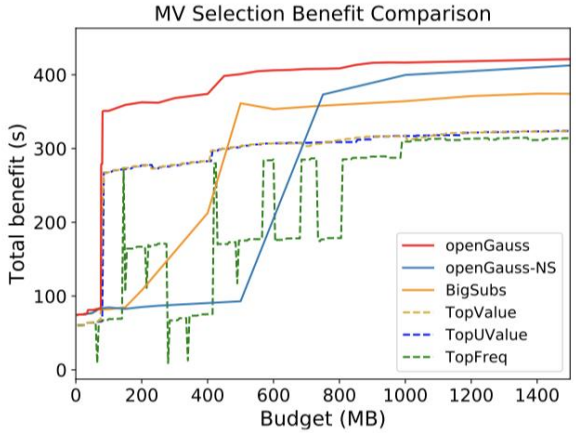


Figure 14: View Advisor – MV Selection (JOB).

Learned Advisor

- Query-embedding: ignoring query plan

Table 7: SQL-Level Diagnosis(4 Datasets)

	Precision	Recall	Latency(ms)
openGauss	**0.913**	**0.922**	0.528
query-embedding	0.739	0.794	**0.035**
PostgreSQL	0.826	0.831	0.372
Actual running	1.0	1.0	3561

Table 6: Anomaly Detection(TPC-C)

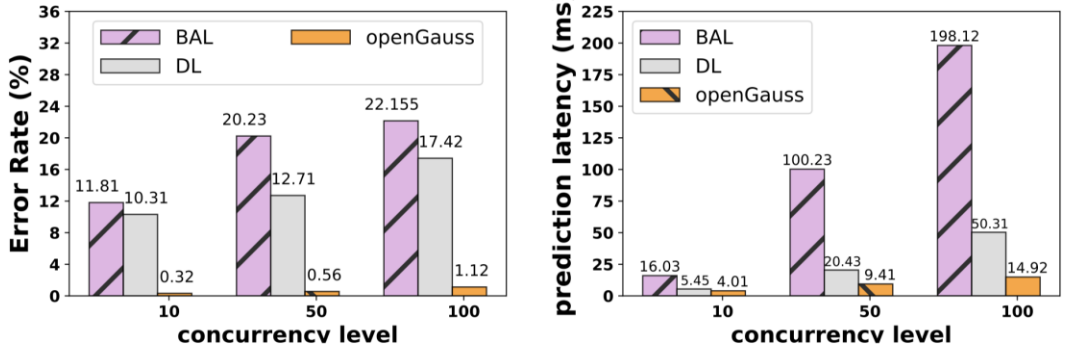
	Precision	Recall	F1-score
openGauss	**0.795**	0.776	**0.785**
VAE	0.302	**0.821**	0.441
GAN	0.554	0.745	0.635

Table 8: System-Level Diagnosis(TPC-C)

	Precision	Recall	F1-score
openGauss	**0.885**	**0.871**	**0.869**
kNN	0.815	0.771	0.765
Decision Tree	0.836	0.824	0.826
DBSherlock	0.826	0.553	0.549

Model Validation

- Outperforms state-of-the-arts methods in prediction accuracy and latency



(a) Mean Square Error (b) Prediction Latency

Figure 16: Model Validation (JOB).

PARTING THOUGHTS

- No forecasting ability
  - Anomaly detection vs workload forecasting
- No end-to-end long-term experiment results
  - Only mentioned knob tuner & index selector results for real-world customers
- How to collect training data for so many models?

References

[1] Guoliang Li, Xuanhe Zhou, Ji Sun, Xiang Yu, Yue Han, Lianyuan Jin, Wenbo Li, Tianqing Wang, Shifu Li. openGauss: An Autonomous Database System. PVLDB, 14(12): 3028 - 3041, 2021. doi:10.14778/3476311.3476380