# QTune: A Query-Aware Database Tuning System with Deep Reinforcement Learning

Paper Authors: Guoliang Li, Xuanhe Zhou, Shifu Li, Bo Gao
Presenter: Jiahao Gai

# Background

*- Knob tuning is an NP-hard problem and existing methods have several limitations.*

1. Limited Scope and Time-Consuming for DBAs manual tuning
2. Dependency on High-Quality Training Data
3. Coarse-Grained Tuning

# Contributions of the paper

1. A query-aware database tuning system using DRL
2. A SQL query featurisation model
3. A DS-DDPG model
4. A DL based query clustering method
5. Experiments on various query workloads and databases outperforming SOTA.

# Architecture

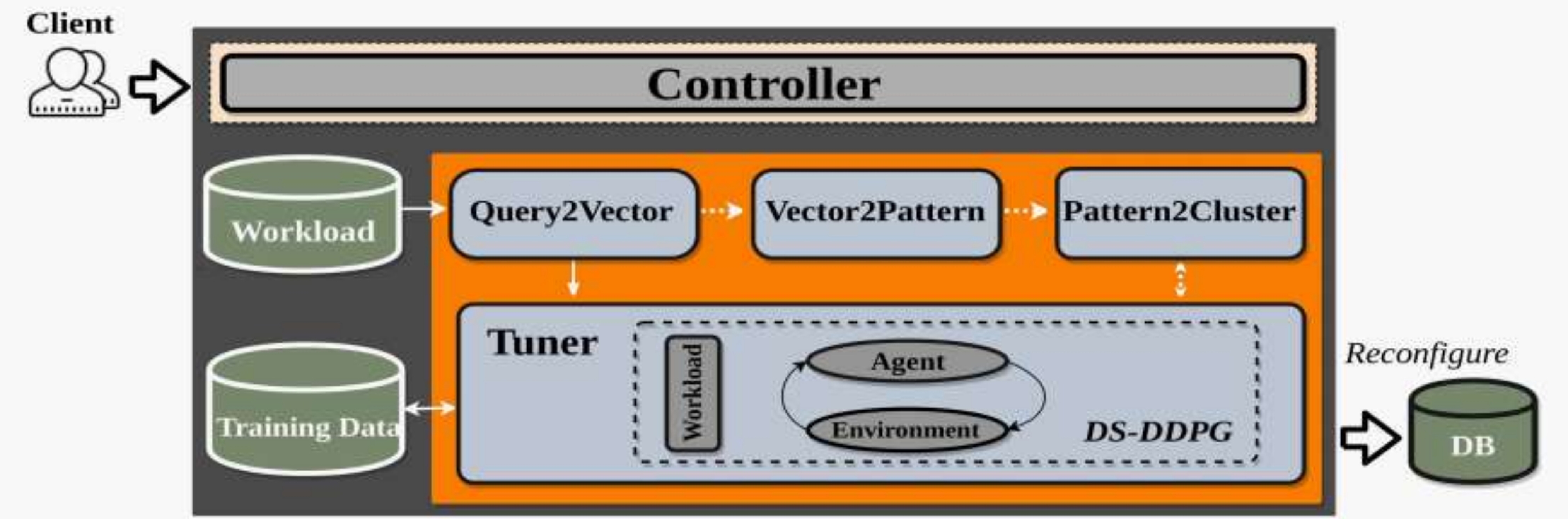



Figure 1: The QTune Architecture

# Workflow

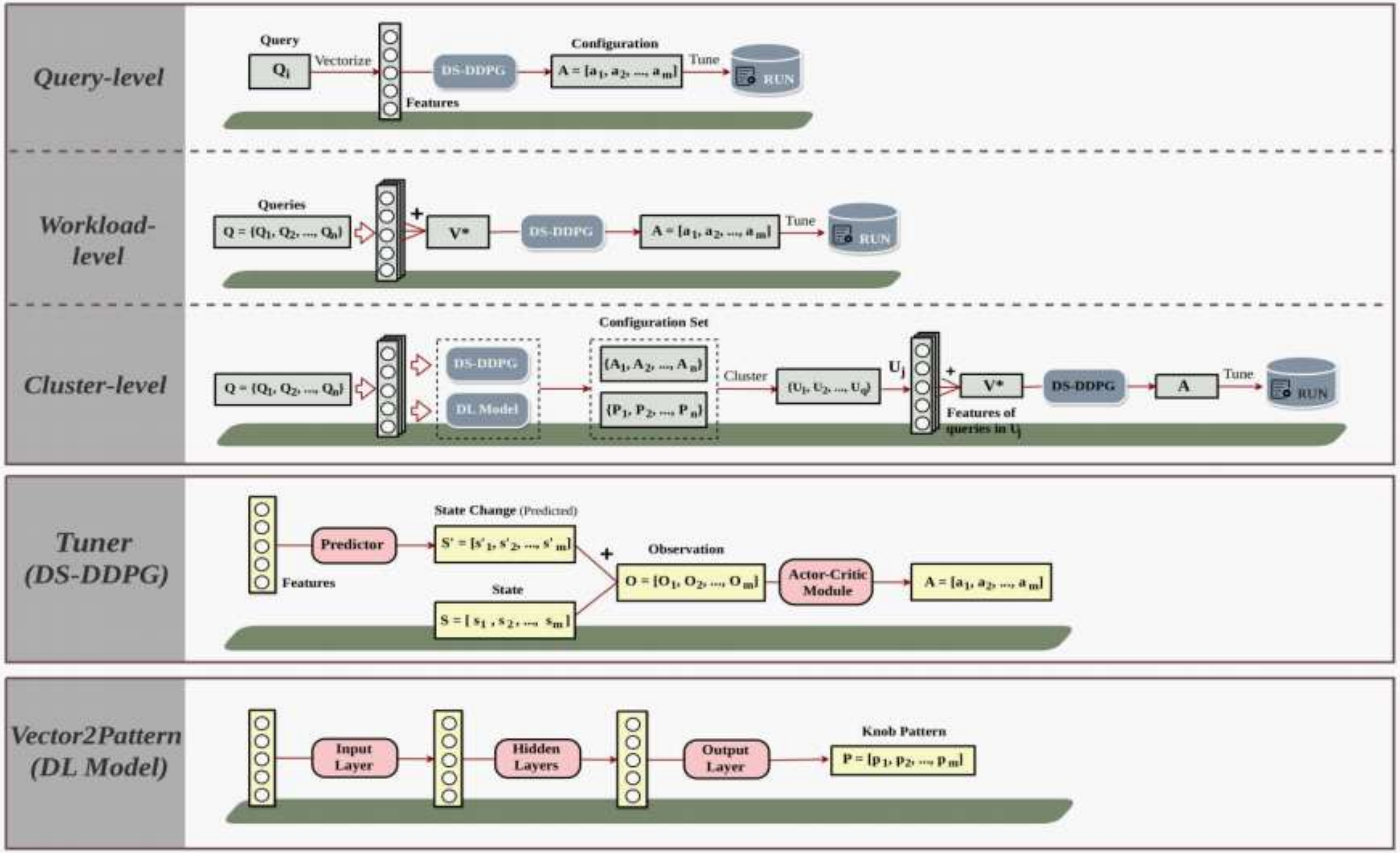



Figure 2: Workflow of QTune

# Query2Vector

1. Query information (0/1)
2. Cost Information (real value)

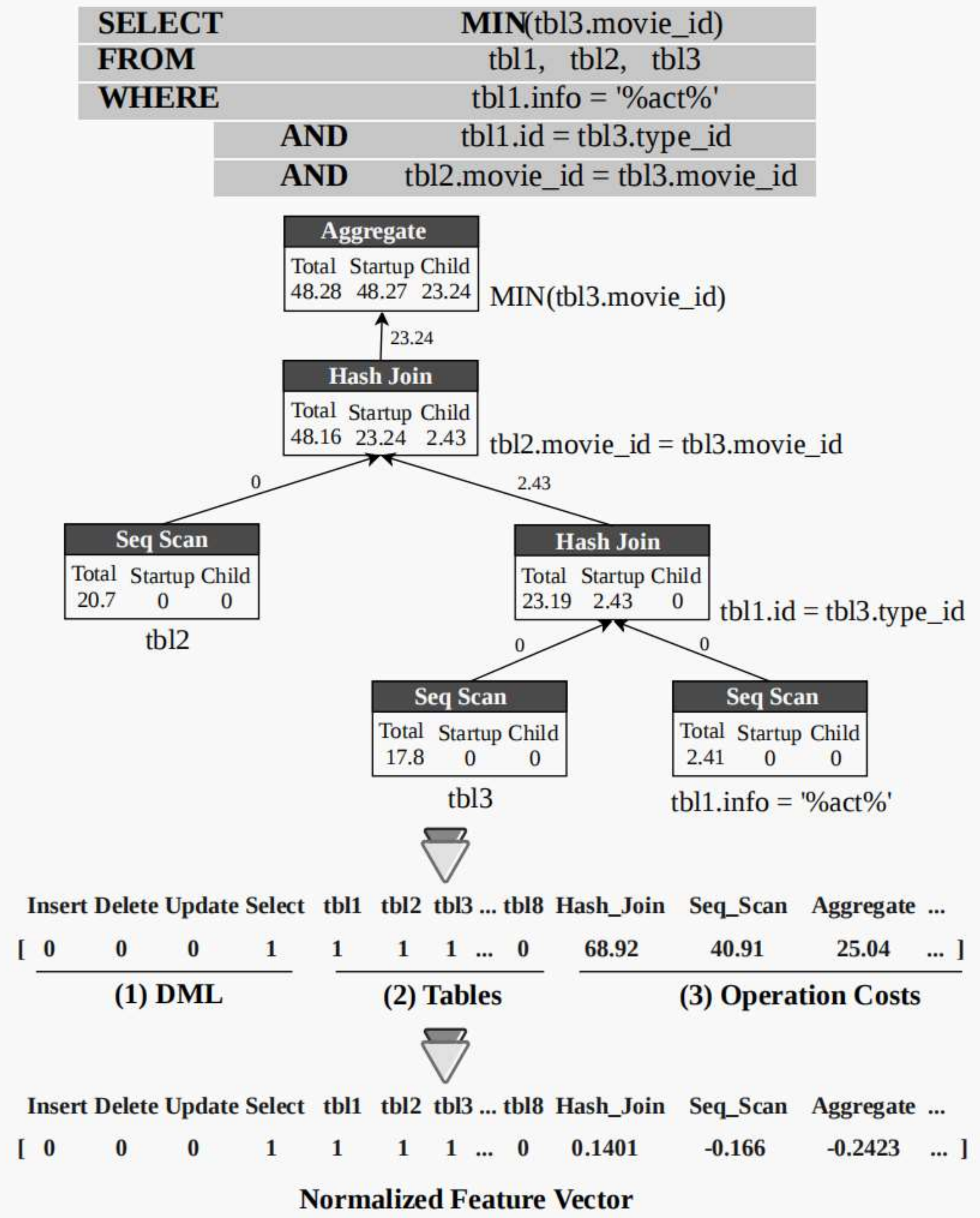



Figure 3: Character Encoding.

# DS-DDPG model

1. Environment
2. Predictor
3. Actor
4. Critic

# Training of DS-DDPG model

1. Training the Predictor
2. Training the Actor-Critic Module

**Algorithm 1:** Training DS-DDPG

**Input:** U: the query set $\{q_1, q_2, \cdots, q_{|U|}\}$
**Output:** $\pi_P$, $\pi_A$, $\pi_C$

Generate training data $T_P$;
TrainPredictor($\pi_P$, $T_P$);
Generate training data $T_A$;
TrainAgent($\pi_A$, $\pi_C$, $T_A$);

# Training the Predictor

**Function** TrainPredictor($\pi_P$, $T_P$)

**Input:** $\pi_P$: The weights of a neural network; $T_P$: The training set

Initiate the weights in $\pi_P$;

**while** *!converged* **do**

  **for** *each* $(v, S, I, \Delta S) \in T_P$ **do**

    Generate the output G of $\langle v, S, I \rangle$;

    Accumulate the backward propagation error: $E = E + \frac{1}{2}||G - \Delta S||^2$;

  Compute gradient $\nabla_{\theta_s}$(E), update weights in $\pi_P$;

# Training the Agent

**Function** TrainAgent($\pi_A$, $\pi_C$, $T_A$)

**Input:** $\pi_A$: The actor's policy; $\pi_C$: The critic's policy; $T_A$: training data

Initialize the actor $\pi_A$ and the critic $\pi_C$;

**while** *!converged* **do**

  Get a training data $T_A^1 = (S_1', A_1, R_1), (S_2', A_2, R_2), \ldots, (S_t', A_t, R_t)$;

  **for** $i = t - 1$ *to* $1$ **do**

    Update the weights in $\pi_A$ with the action-value $Q(S_i', A_i|\pi_C)$;

    Estimate an action-value $Y_i = R_i + \tau Q(S_{i+1}', \pi_A(S_{i+1}'|\theta^{\pi A})|\pi_C)$;

    Update the weights in $\pi_C$ by minimizing the loss value $L = (Q(S_i', A_t|\pi_C) - Y_i)^2$;

# Granularities of tuning

1. **Query-level:** can optimise the query latency; but low throughput
2. **Workload-level:** cannot optimise the query latency; high throughput
3. **Cluster-level:** can optimise both the latency and throughput

# Granularities of tuning

1. **Query-level:** can optimise the query latency; but low throughput

2. **Workload-level:** cannot optimise the query latency; high throughput

3. **Cluster-level:** can optimise both the latency and throughput

# Evaluation

1. Three query workloads JOB, TPC-H and Sysbench

Table 3: Workloads. RO, RW and WO denote read-only, read-write and write-only respectively.

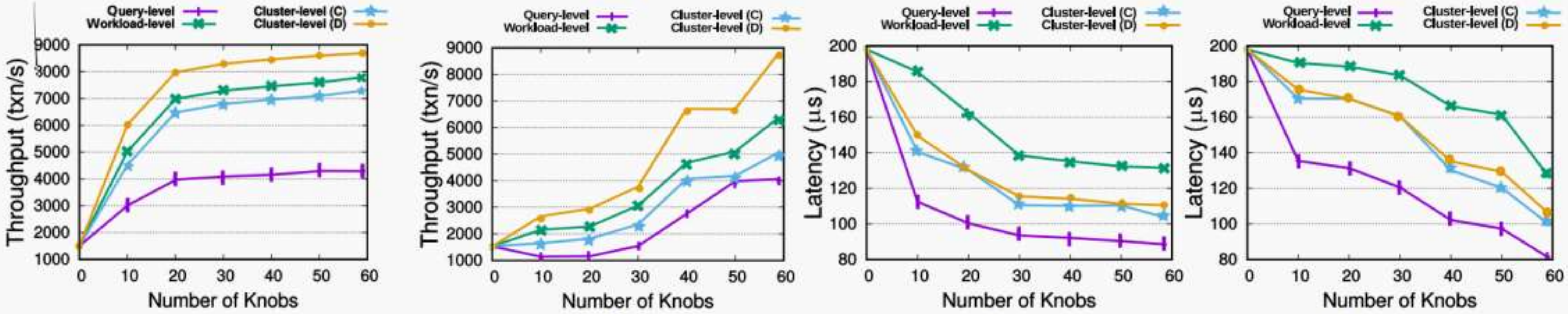
| Name | Mode | Table | Cardinality | Size(G) | Query |
|---|---|---|---|---|---|
| JOB | RO | 21 | 74,190,187 | 13.1 | 113 |
| TPC-H | RO | 8 | 158,157,939 | 50.0 | 22 |
| Sysbench | RO, RW | 3 | 4,000,000 | 11.5 | 474,000 |

2. Metrics: latency, throughput, as well as the training and tuning time
3. Three kinds of databases

Table 2: Database information

| Database | Knobs without restart | State Metrics |
|---|---|---|
| PostgreSQL | 64 | 19 |
| MySQL | 260 | 63 |
| MongoDB | 70 | 515 |

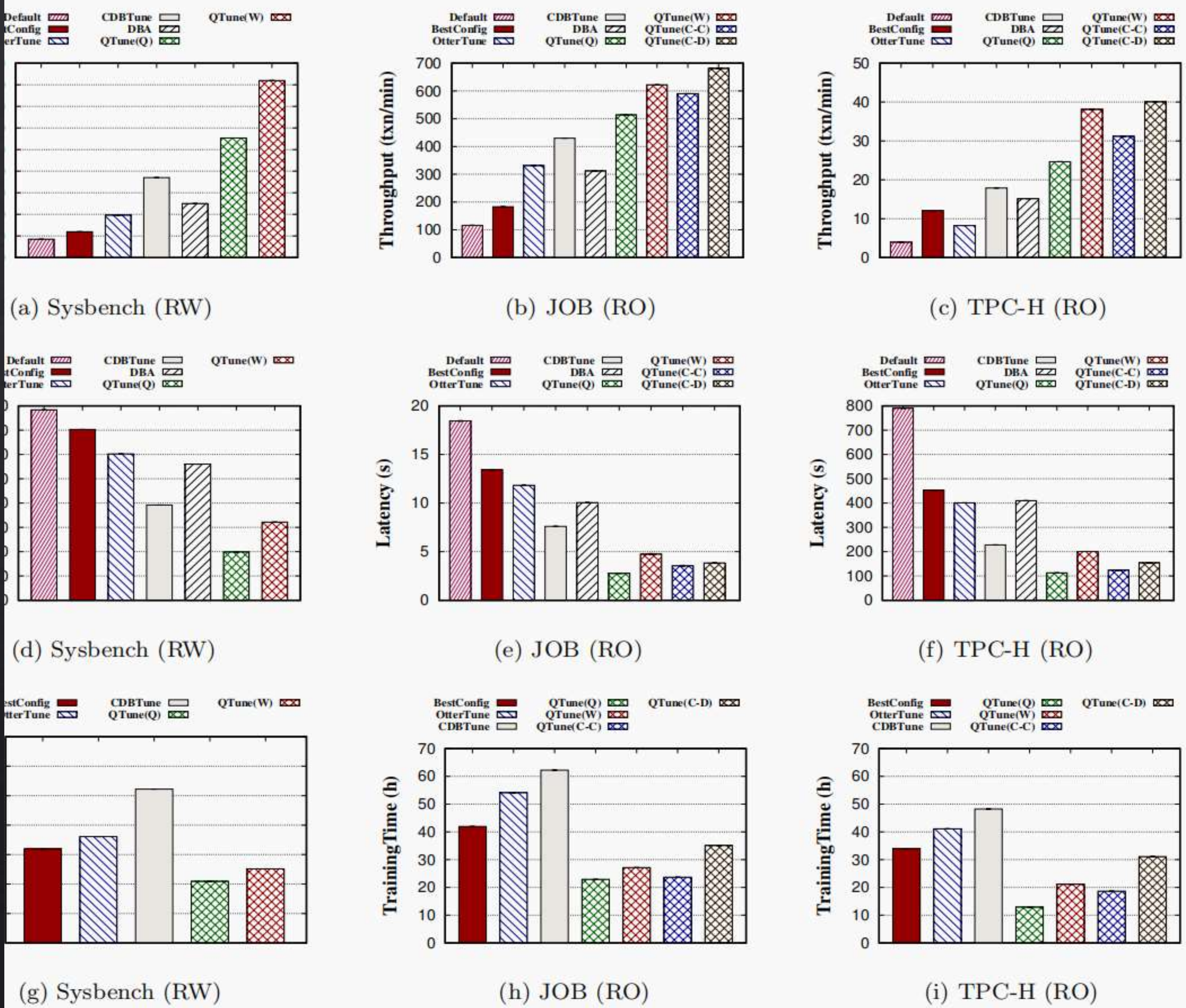
# Evaluating three types of tuning




(a) IF-Throughput (b) RC-Throughput (c) IF-Latency (d) RC-Latency

**Figure 6: Performance by increasing knobs in Important First (IF) and Randomly Choosing (RC) respectively when running Sysbench (RO) on PostgreSQL.**

# Comparison with Existing Techniques



# Reviews

**Pros**

- Comprehensive evaluations

**Cons**

- Feature vectorisation makes the database hard to add and delete future tables

- It is unclear why they did not provide cluster-level evaluations on the Sysbench dataset

# Questions?