Lecture #07

Carnegie Mellon University

*Special Topics:*

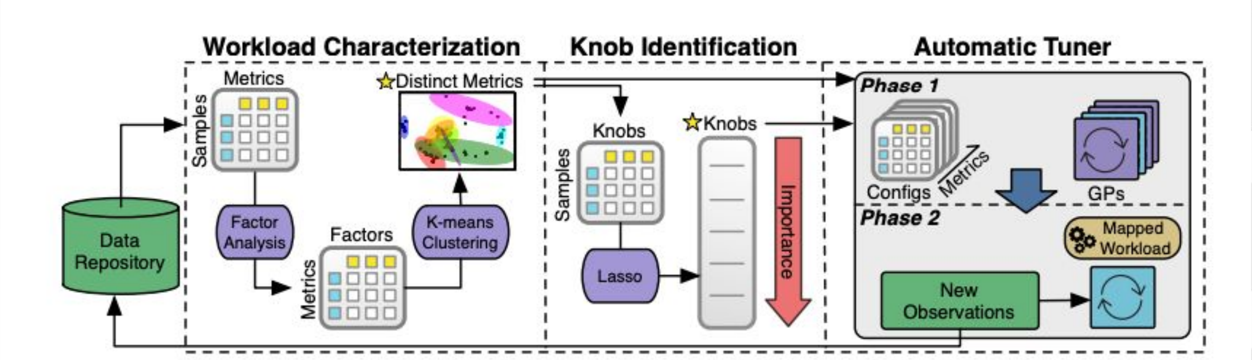
# Self-Driving Database Management Systems

## Knob/Parameter Tuning II

Neville Chima // 15-799 // Spring 2022

# LAST CLASS - OTTERTUNE

- Select most impactful knobs
- Map new workloads to previous workloads
- Recommend knob settings






MAKE YOUR DATABASE SYSTEM DREAM OF ELECTRIC SHEEP: TOWARDS SELF-DRIVING OPERATION
*VLDB 2021*

# TODAY'S AGENDA

- Overview
- System Architecture
- Methods
- Evaluation
- Thoughts



Source: Lin Ma

# MOTIVATION

**Problem:** DBAs expertise do not suffice in tuning knob configuration for DBMSs

**Goal**: Develop efficient system for automatic optimization of knob configuration (in CDBs)

- Class of system ?
- Capabilities of system ?

# EXISTING FRAMEWORK

## Search Based Methods e.g Bestconfig

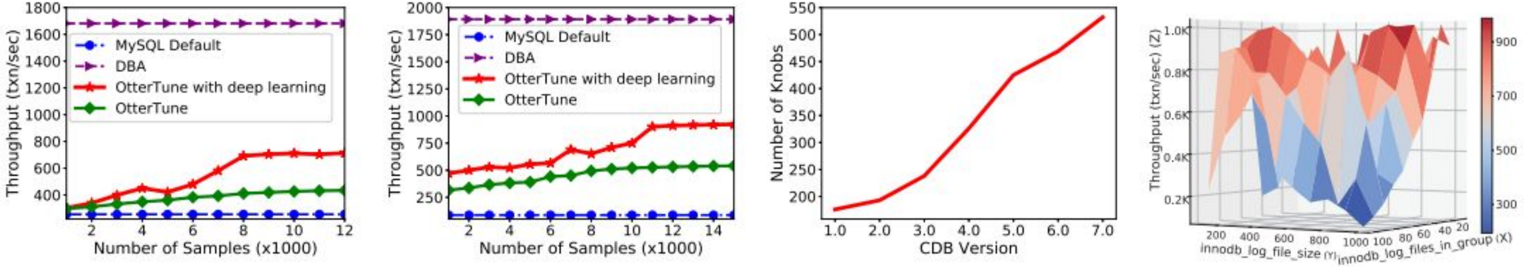

- Heuristic search

## Learning Based Methods e.g Ottertune

- ML on historical data

# CHALLENGES

1. Time Consuming - **SB**
2. Inability to optimize overall performance - **SB, LB**
3. Performance in a cloud environment - **LB**
4. High dimensional knob space - **LB**




(a) CDB (TPC-H) (b) CDB (Sysbench) (c) Knobs Increase (d) Performance surface

# TODAY'S AGENDA

- Overview
- System Architecture
- Methods
- Evaluation
- Thoughts



Source: Lin Ma

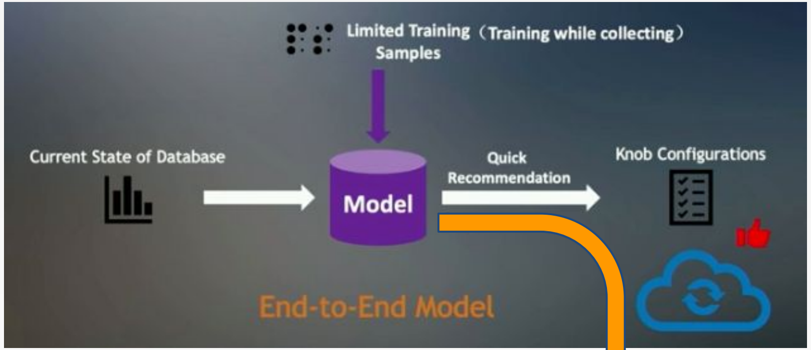
# TODAY'S AGENDA

- Overview
- System Architecture
  - Components
  - System Mechanism
- Methods
- Evaluation
- Thoughts



Source: Lin Ma

# SYS ARCH - COMPONENTS




- Simulates standard workload
- Replay User workload

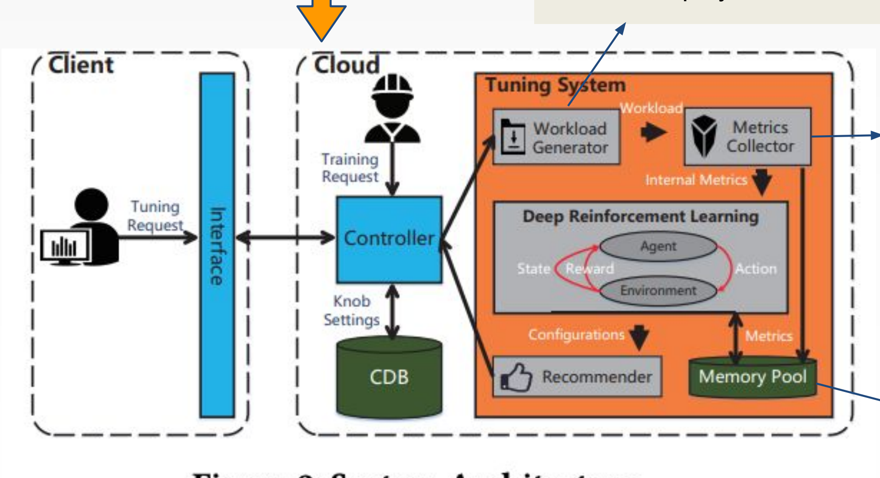



Figure 2: System Architecture.

- Retrieve Internal Metrics
- Sample for External metrics
- Average, Cumulative, Difference values

- Stores training samples ( <s,r,a,s> transitions)

# System Mechanism

- Offline training
  - Bootstrapped cold start
  - Reinforcement Learning (RL) exploration
- Online tuning
  - Incremental training on user data
  - Updates to RL model & Memory pool

# TODAY'S AGENDA

- Overview
- System Architecture
- Methods
- Evaluation
- Thoughts



Source: Lin Ma

# TODAY'S AGENDA

- Overview
- System Architecture
- Methods
  - Deep RL
  - Reward Selection
- Evaluation
- Thoughts



Source: Lin Ma

# RL - INSPIRATION

*"Abstract tuning problem into a scoring game"*

**Rule:** Tune knobs at regular intervals and obtain each performance


**Reward:** Based off a reward function

- Performance enhancement - +ve reward value
- Performance degradation - -ve reward value

**Goal:** Ultimately achieve a higher expected reward within a few tries (exploration vs exploitation) as possible

# RL IN CDBTUNE

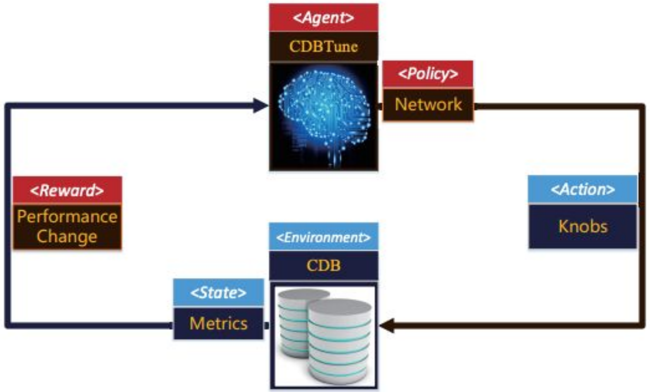
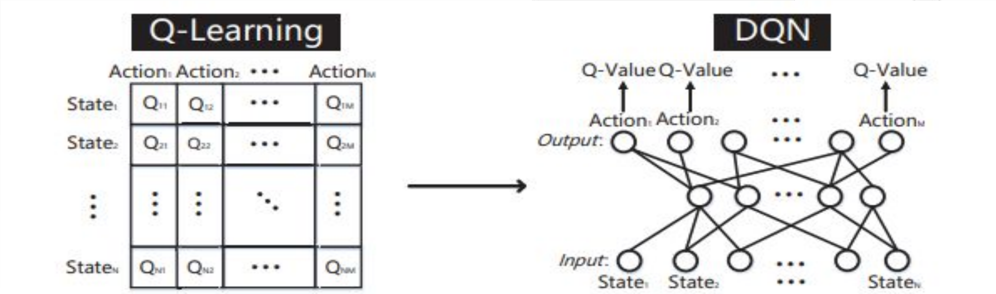



**Figure 3: The correspondence between RL elements and CDB configuration tuning.**

| | |
|---|---|
| Agent | CDBTune<br>receives reward<br>updates policy for exp reward |
| Environment | tuning target - CDB instance |
| State $s_t$ | Internal metrics<br>Track state of the env |
| Reward $r_t$ | Change in performance after applying recs |
| Action $a_t$ | Knob Tuning operation<br>Given policy and state of CDB |
| Policy $\mu(s_t)$ | Behaviour of CDBTune given time & env - RL network |

# RL - CONSIDERATIONS

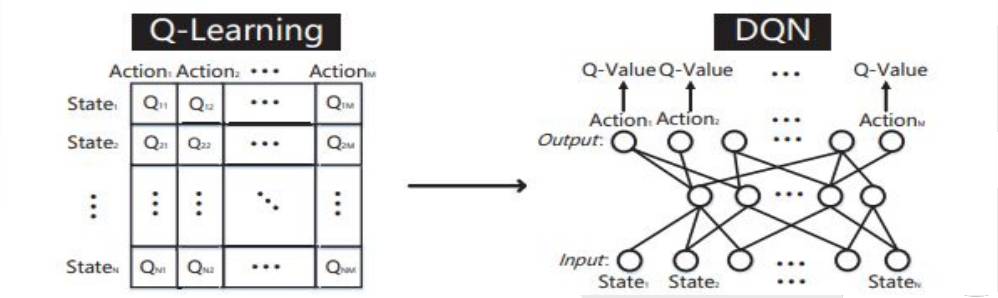
- Q-learning
  - Calculation of Q-state tables
  - $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r + \gamma max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (1)$$
- Deep Q Networks
  - Neural networks to calculate Q-values (benefit of action)
  - $$Q(s, a, \omega) \rightarrow Q(s, a)$$



# RL - CONSIDERATIONS

- Q-learning
  - Calculation of Q-state tables
  - $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r + \gamma max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (1)$$
- Deep Q Networks
  - Neural networks to calculate Q-values (benefit of action)
  - $$Q(s, a, \omega) \rightarrow Q(s, a)$$

**Continuous high dimensionality space ?!**



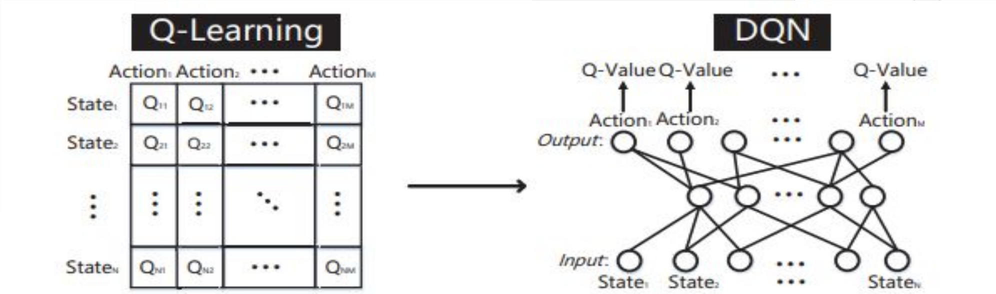
# RL - CONSIDERATIONS

- Q-learning
  - Calculation of Q-state tables
  - $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r + \gamma max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (1)$$

**Unfeasible for large state spaces**

- Deep Q Networks
  - Neural networks to calculate Q-values (benefit of action)
  - $$Q(s, a, \omega) \rightarrow Q(s, a)$$

**Discrete-controlled outputs**



# RL - DDPG

- Deep Deterministic Policy Gradient
  - DQN
  - Actor-Critic Algorithm
    - Actor: produces probability value for each action in the knob space
    - Critic: estimates sum of future rewards
  - Acquire single Q-value for current action & state

# DDPG ALGORITHM

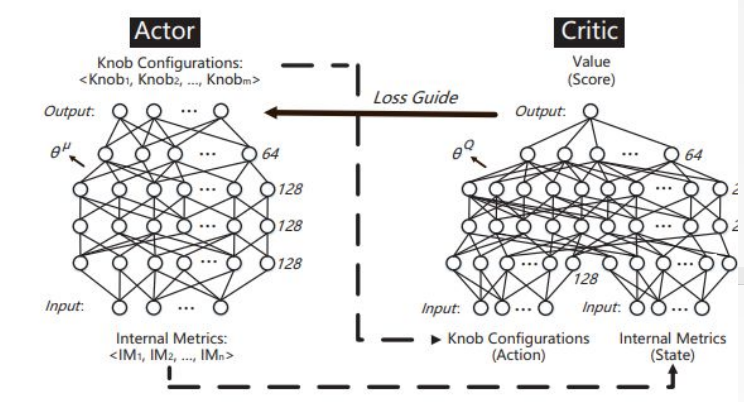
**Algorithm 1** Deep deterministic policy gradient (DDPG)

1: Sample a transition $(s_t, r_t, a_t, s_{t+1})$ from Experience Replay Memory.
2: Calculate the action for state $s_{t+1}$: $a'_{t+1} = \mu(s_{t+1})$.
3: Calculate the value for state $s_{t+1}$ and $a'_{t+1}$: $V_{t+1} = Q(s_{t+1}, a'_{t+1}|\theta^Q)$.
4: Apply Q-learning and obtain the estimated value for state $s_t$: $V'_t = \gamma V_{t+1} + r_t$.
5: Calculate the value for state $s_t$ directly: $V_t = Q(s_t, a_t|\theta^Q)$.
6: Update the critic network by gradient descent and define the loss as:

$$L_t = (V_t - V'_t)^2$$

7: Update the actor network by policy gradient:

$$\begin{aligned}\nabla_{\theta^\mu} J &\approx \mathbb{E}[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)}] \\ &= \mathbb{E}[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_t}]\end{aligned}$$




| | | |
|---|---|---|
| $\mu$ | Policy | deep neural network |
| $\theta^Q$ | Learnable parameters | Initialized to *Normal*(0, 0.01) |
| $\theta^\mu$ | Actor, mapping state $s_t$ to action $a_t$ | - |
| $Q^\mu$ | Critic, the policy $\mu$ | - |
| $L$ | Loss function | - |
| $y$ | Q value label through Q-learning algorithm | - |

# REWARD SELECTION - IDEA

- Initial performance before tuning is $D_0$
- After the i-th tuning operation $D_i$ , DBA compares performance btw i) $D_i$ and $D_{i-1}$ ii) $D_i$ and $D_{0.}$
- If $D_i$ is better than $D_0$, tuning trend is correct & reward is positive, else negative.
- Thus, reward is modeled considering $\Delta(D_i , D_0)$ & $\Delta(D_i, D_{i-1})$

# REWARD SELECTION - DETAILS

$$r = \begin{cases} ((1+\Delta_{t\to 0})^2 - 1)|1+\Delta_{t\to t-1}|, \Delta_{t\to 0} > 0 \\ -((1-\Delta_{t\to 0})^2 - 1)|1-\Delta_{t\to t-1}|, \Delta_{t\to 0} \leqslant 0 \end{cases}$$

$$r = C_T * r_T + C_L * r_L$$

$$C_L + C_T = 1.$$

$$\Delta T = \begin{cases} \Delta T_{t\to 0} = \dfrac{T_t - T_0}{T_0} \\ \Delta T_{t\to t-1} = \dfrac{T_t - T_{t-1}}{T_{t-1}} \end{cases}$$

$$\Delta L = \begin{cases} \Delta L_{t\to 0} = \dfrac{-L_t + L_0}{L_0} \\ \Delta L_{t\to t-1} = \dfrac{-L_t + L_{t-1}}{L_{t-1}} \end{cases}$$

$T_t$ : *Throughput (txn/sec)at time t*
$L_t$: *Latency (ms) at time t*
$r$: *Reward*
$C_l$: *Coefficient of latency*
$C_t$: *Coefficient of throughput*

# TODAY'S AGENDA

- Overview
- System Architecture
- Methods
- Evaluation
- Thoughts



Source: Lin Ma

# TODAY'S AGENDA

- Overview
- System Architecture
- Methods
- Evaluation
  - Execution time
  - Baseline comparisons
- Thoughts



Source: Lin Ma

# EXECUTION TIME

- Offline Training: 4.7 hrs for **266** knobs, **2.3** hours for **65**
- Online Tuning: **5** steps in **25** min
- Step time division:

| Step | Time Taken |
|---|---|
| Stress Testing | 153s |
| Metrics collection | 0.86 ms |
| Model update | 28.76 ms |
| Recommendation calculation | 2.16 ms |
| Deployment time | 17 s |

# EFFICIENCY COMPARISON

- Experiments on CDB-A instance
- CDBTune takes shorter tuning time
- CDBTune gradually adapts to workload as tuning steps increase
  - Initially already achieves better results than other DBA, Ottertune, & BestConfig

**Table 2: Detailed online tuning steps and time of CDBTune and other tools.**

| Tuning Tools | Total Steps | Time of One Step (mins) | Total Time (mins) |
|---|---|---|---|
| CDBTune | 5 | 5 | 25 |
| OtterTune | 5 | 11 | 55 |
| BestConfig | 50 | 5 | 250 |
| DBA | 1 | 516 | 516 |

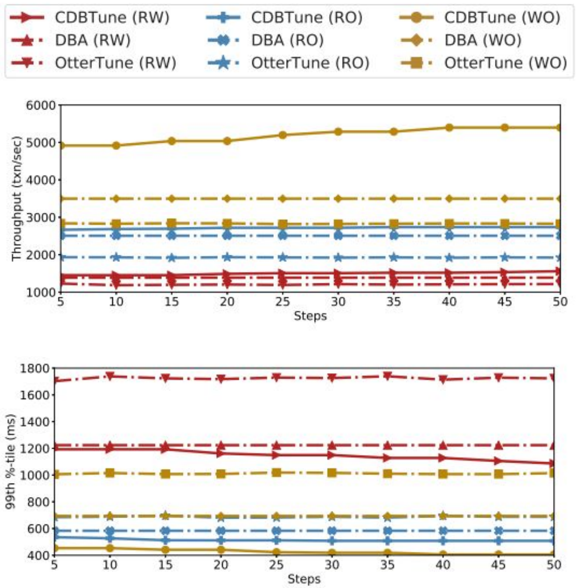
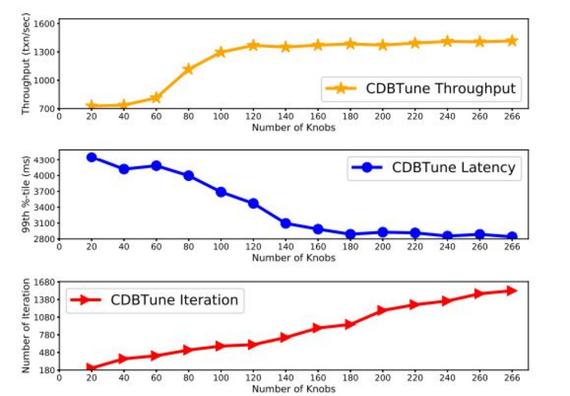



**Figure 5: Performance by increasing number of steps**

# EFFECTIVENESS COMPARISON

- Experiments on CDB-B instance
- DBA & Ottertune’s knob ordering
- CDBTune maintains better performance as knob space increases
  - Dependencies in larger knob spaces
- Stability as knob space increases
  - Abstracted knob ranking via features in NN



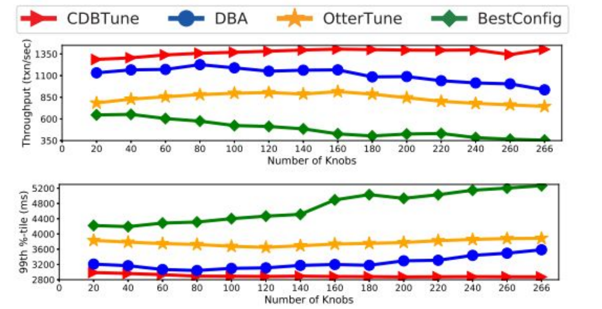




**Figure 6: Performance by increasing number of knobs (knobs sorted by DBA).**

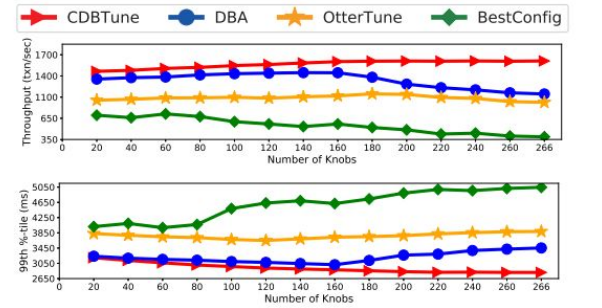



**Figure 7: Performance by increasing number of knobs (knobs sorted by OtterTune).**

# ADAPTABILITY COMPARISONS

## Identical performance btw normal & cross-testing

Training on Model

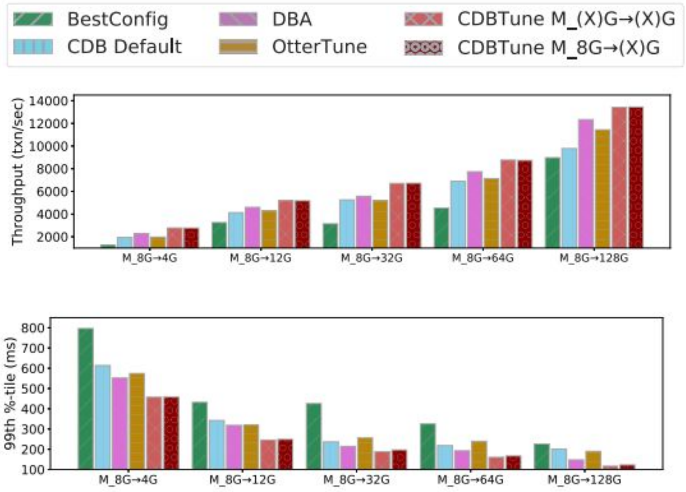



**Figure 10: Performance comparison for Sysbench WO workload when applying the model trained on 8G memory to (X)G memory hardware environment.**

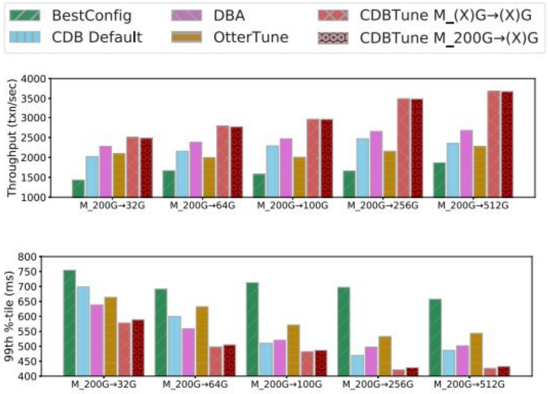



**Figure 11: Performance comparison for Sysbench RO workload when applying the model trained on 200G disk to (X)G disk hardware environment.**

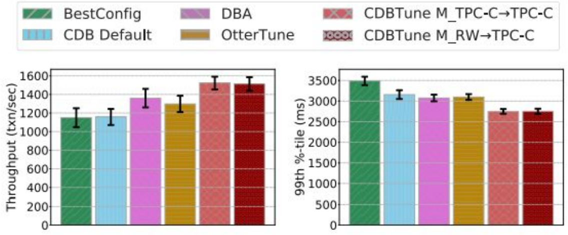



**Figure 12: Performance comparison when applying the model trained on Sysbench RW workloads to TPC-C.**

# PARTING THOUGHTS

- Limited samples
- Reduces possibility of local optimum
- Good adaptability
- Applies to high-dimensional continuous knob space
- End to end approach ?
  - Connectivity of HL
  - Multi-Model Approach

# NEXT CLASS

Knob/Parameter Tuning III