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
TODAY’S AGENDA

- Overview
- System Architecture
- Methods
- Evaluation
- Thoughts
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
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
- Retrieve Internal Metrics
- Sample for External metrics
- Average, Cumulative, Difference values
- Stores training samples (\(<s,r,a,s>\) transitions)

Figure 2: System Architecture.
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

![Diagram](image)

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

<table>
<thead>
<tr>
<th><strong>Agent</strong></th>
<th><strong>CDBTune receives reward</strong>&lt;br&gt;<strong>updates policy for exp reward</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Environment</strong></td>
<td><strong>tuning target - CDB instance</strong></td>
</tr>
<tr>
<td><strong>State</strong> $s_t$</td>
<td><strong>Internal metrics</strong>&lt;br&gt;<strong>Track state of the env</strong></td>
</tr>
<tr>
<td><strong>Reward</strong> $r_t$</td>
<td><strong>Change in performance after applying recs</strong></td>
</tr>
<tr>
<td><strong>Action</strong> $a_t$</td>
<td><strong>Knob Tuning operation</strong>&lt;br&gt;<strong>Given policy and state of CDB</strong></td>
</tr>
<tr>
<td><strong>Policy</strong> $\mu(s_t)$</td>
<td><strong>Behaviour of CDBTune given time &amp; env - RL network</strong></td>
</tr>
</tbody>
</table>
RL - CONSIDERATIONS

- **Q-learning**
  - Calculation of Q-state tables
  - \[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \]  
  
- **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
    \[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \]  
  - \( Q(s, a, \omega) \rightarrow Q(s, a) \)

- **Deep Q Networks**
  - Neural networks to calculate Q-values (benefit of action)
    - Continuous high dimensionality space ?!
**RL - CONSIDERATIONS**

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

- **Deep Q Networks**
  - Neural networks to calculate Q-values (benefit of action)
  - $Q(s, a, \omega) \rightarrow Q(s, a)$
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
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 \(s_t\): \(V'_t = y V_{t+1} + r_t\).
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:
\[
\nabla_{\theta^\mu} J \approx \mathbb{E}[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)}] \\
= \mathbb{E}[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_t}]
\]

<table>
<thead>
<tr>
<th>(\mu)</th>
<th>Policy</th>
<th>deep neural network</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\theta^Q)</td>
<td>Learnable parameters</td>
<td>Initialized to Normal(0, 0.01)</td>
</tr>
<tr>
<td>(\theta^\mu)</td>
<td>Actor, mapping state (s_t) to action (a_t)</td>
<td>-</td>
</tr>
<tr>
<td>(Q^\mu)</td>
<td>Critic, the policy (\mu)</td>
<td>-</td>
</tr>
<tr>
<td>(L)</td>
<td>Loss function</td>
<td>-</td>
</tr>
<tr>
<td>(y)</td>
<td>Q value label through Q-learning algorithm</td>
<td>-</td>
</tr>
</tbody>
</table>
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

\[ r = \begin{cases} 
( (1 + \Delta_{t \rightarrow 0})^2 - 1 ) |1 + \Delta_{t \rightarrow t-1}|, & \Delta_{t \rightarrow 0} > 0 \\
-((1 - \Delta_{t \rightarrow 0})^2 - 1 ) |1 - \Delta_{t \rightarrow t-1}|, & \Delta_{t \rightarrow 0} \leq 0 
\end{cases} \]

\[ r = C_T \cdot r_T + C_L \cdot r_L \]

\[ C_L + C_T = 1. \]

\[ \Delta T = \begin{cases} 
\Delta T_{t \rightarrow 0} = & \frac{T_t - T_0}{T_0} \\
\Delta T_{t \rightarrow t-1} = & \frac{T_t - T_{t-1}}{T_{t-1}} 
\end{cases} \]

\[ \Delta L = \begin{cases} 
\Delta L_{t \rightarrow 0} = & \frac{-L_t + L_0}{L_0} \\
\Delta L_{t \rightarrow t-1} = & \frac{-L_t + L_{t-1}}{L_{t-1}} 
\end{cases} \]

\( T_t \): Throughput (txn/sec) at time \( t \)
\( L_t \): Latency (ms) at time \( t \)
\( r \): Reward
\( C_t \): 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
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:

<table>
<thead>
<tr>
<th>Step</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stress Testing</td>
<td>153s</td>
</tr>
<tr>
<td>Metrics collection</td>
<td>0.86 ms</td>
</tr>
<tr>
<td>Model update</td>
<td>28.76 ms</td>
</tr>
<tr>
<td>Recommendation calculation</td>
<td>2.16 ms</td>
</tr>
<tr>
<td>Deployment time</td>
<td>17 s</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Tuning Tools</th>
<th>Total Steps</th>
<th>Time of One Step (mins)</th>
<th>Total Time (mins)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDBTune</td>
<td>5</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>OtterTune</td>
<td>5</td>
<td>11</td>
<td>55</td>
</tr>
<tr>
<td>BestConfig</td>
<td>50</td>
<td>5</td>
<td>250</td>
</tr>
<tr>
<td>DBA</td>
<td>1</td>
<td>516</td>
<td>516</td>
</tr>
</tbody>
</table>

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