BLUTune: Continuous Knobs Tuning

Canadian AI 🍁 2021

Spencer Bryson, Connor Henderson, Parke Godfrey, Jarek Szlichta
Mohammed Alhamid, Vincent Corvinelli, Piotr Mierzejewski, Calisto Zuzarte

IBM
York University
Ontario Tech University
What is Knob Tuning?

• DBMS have **dozens of knobs** (configuration parameters) that control them
  – e.g. Sort heap size, buffer pool size, optimization levels, concurrency control, etc

• Knobs must be properly adjusted to achieve high performance and scalability
  – high throughput and low latency
Motivation

• Traditionally, databases rely on DBAs to tune the knobs
  – Non-trivial problem
  – Too many knobs
  – Requires an expert to spend a lot of time and effort (possibly several days)

• As a workload evolves over time the configuration may no longer be optimal
  – Could cause poor performance until the knobs are re-tuned
Automatic Knob Tuning

• A fully automated approach to optimally tune knobs is desired
  – Relieve the burden of tuning from experts
  – Find better configurations than experts

• Businesses and their applications are not static
  – Workloads can change overtime; so must the knobs
  – Cannot “set it and forget it”
Challenges and Use Cases

• Many knobs are **continuous** values

• Knobs can be **interdependent**

• One configuration **does not fit all**
Our Approach Architecture

- An intelligent ML solution driven by deep reinforcement learning
  - We use actor-critic network with policy-based learning to compute most likely best next action
  - embeddings to map high-dimensional queries into low-dimensional representation
Reward-driven learning

- The agent’s behaviour is driven by the designed reward function
  - Goal is to maximize the reward

- Our reward is based on change in query performance
  - Performance metric can be execution time or optimizer cost
  - We keep a cost history and apply exponential decay to steer the learning

- Our reward function also enforces limited resource constraints
  - i.e. the environment only has X amount of shared memory (sortheap, bpsize) e.g., cloud computing, must train the agent to stay within constraints
Execution time vs. Optimizer cost

• The overall goal of the agent is to **minimize the execution time** by finding a suitable configuration
  – Training on **execution time** can get **prohibitively expensive** as the complexity of the queries and the size of the database grows
    • The size of the database for prior works is mostly from 1GB to 10GB (we target 100GB+ large & complex query workloads)

• We’ve demonstrated that **optimizer cost** can be used in lieu of execution time
  – greatly speeds up training
  – **20 episodes in 7 hours** vs. **5 episodes in 38 hours**
Fine-tuning the model

• **Optimizer cost** can be an effective measure of performance for various knob changes

• However, **execution time** captures some information that the optimizer fails with (inaccurate estimates, knob not factored in, etc)

• Thus, it is desirable to use the concept of **transfer learning**
  – first train up a model on minimizing optimizer cost, and then **fine-tune** the model by training on minimizing execution time
Results 1: Cost-only model

- Trained ONLY on cost
- Time taken: \(~3.73\) hours for 10 episodes (1000 iterations each)
Results 2: Fine-tuned on Execution time

- Continue training on the cost-only model with execution time
- **Time taken:** ~13.7 hours for 4 episodes (500 iterations each)
Result comparison

• Training only on cost produces a configuration that results in a total execution time of 107.91 seconds

• Fine-tuning the model above (transfer learning) leads to a execution time of 93 seconds

• This is an additional ~15% decrease which illustrates that fine-tuning the transfer learning component is beneficial
Thank-you