Deep Reinforcement Learning in Computer Systems: Learning from Traces

Michael Schaarschmidt, Eiko Yoneki,
firstname.lastname@cl.cam.ac.uk

Managing efficient configurations is a central challenge in computer systems. For example, database systems expose a large variety of configuration parameters to customize every aspect of data management from storage engine to high level request parameters. Efficient deployments require optimized resource management, task scheduling, and fine-tuned configurations.

Parameters are typically determined through a combination of manual work, software vendors, and open source community recommendations. Recently, auto-tuning tools such as Spearmint or OpenTuner [2, 1] have been integrated into this process. Users employ such tools by exposing configuration parameters and application performance metrics to a black box optimizer which iteratively improves and evaluates configurations. A popular auto-tuning technique is Bayesian optimization, which incrementally builds a probabilistic model of parameter performance. These *offline* methods produce *static* configurations representing the best compromise between different service demands of an application.

In contrast, in the *online* case, distinct workloads are pre-identified offline and then matched to different configurations online based on continuous monitoring. Designing such solutions frequently leads to a constructive understanding of system and workload behaviour, and the resulting solutions tend to be well understood. In turn, such tailor-made models require extensive human analysis and experimentation, and may lack portability. Reinforcement learning (RL) suggests to address this problem by learning an adaptive control policy by trial and error without an analytical model [3]. Recent successes in combining neural networks with reinforcement learning have sparked new interested in attempting to learn adaptive configuration policies from raw system data. While theoretically appealing, practical deployments of reinforcement learning solutions remain elusive due to large training data requirements, algorithmic instability, and lack of standard tools.

RL algorithms do not require supervised training data but need to interact with their environment to explore the state space (initially sampling random actions), and to learn about the impact of decisions. The theoretical ability to continuously learn from new experiences is difficult to realize in practice if behavior in new situations is unpredictable and may have catastrophic consequences. A common strategy to avoid exposing production systems to unpredictable behavior is to train RL agents in simulation and deploy the simulation-trained model. This approach enables researchers to explore small proof-of-concept experiments but also introduces the risk of making unrealistic assumptions and oversimplifying the problem domain. Some research domains (e.g. networking) have access to reliable protocol simulators but this is not the case for many optimization problems.

This paper investigates mechanisms to close the gap between research and practice via both software infrastructure and algorithms. We address the issue of instability and data efficiency by focusing on systems with rich performance profiling. We further introduce TensorForce[1], a deep reinforcement learning library for applied use cases focusing on strict separation of environment and RL model. Our key practical insight is that log data of *static configurations* can be leveraged to guide model creation without requiring a simulator. We evaluate the end-to-end approach for generating controllers on a proof-of-concept case study learning compound indexing from database profiling logs. Results show that logs of static configurations can be used to generate a dynamic controller for automatic indexing.

References

[1] Ansel, Jason et al. Opentuner: An extensible framework for program autotuning. PACT '14, pages 303–316, New York, NY, USA, 2014. ACM.

[2] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *NIPS*. 2012.

[3] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[1] https://github.com/reinforceio/tensorforce

UNIVERSITY OF CAMBRIDGE

Deep Reinforcement Learning in Computer Systems: Learning from Traces

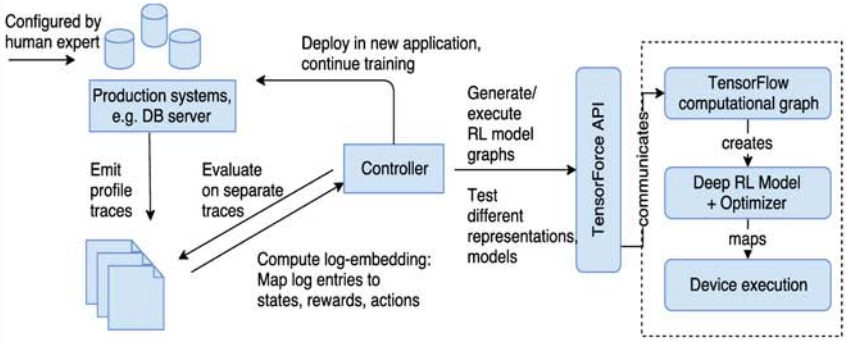
Michael Schaarschmidt, Eiko Yoneki
Department of Computer Science and Technology, University of Cambridge
(E-mail: firstname.lastname@cl.cam.ac.uk)

Problem:

Controlling dynamic behavior In computer systems:

- Complex configuration parameter space
- Increasing number of parameters
- Hand-crafted solutions impractical, often left static or configured through extensive offline analysis
- Reinforcement learning algorithms require extensive online training or simulations

Example: Dynamic Database Indexing



Concept:

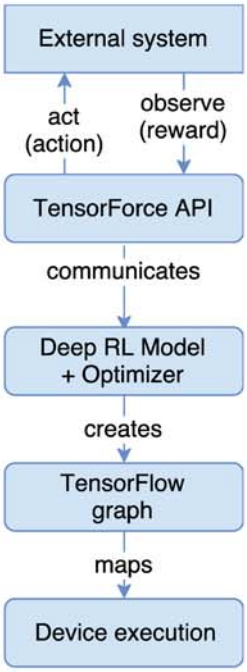
Leverage existing trace data to guide model creation:

- Use static examples of expert configured system, e.g. database tables configured by administrator
- Map log entries to workload embedding to present to agent, e.g. query structure, unique queries, frequency
- Interpret as supervised examples of actions to take given a set of queries

A software stack for RL in systems

Various tuning packages exist for static parameter tuning, e.g. OpenTuner, Spearmint, BOAT but no similar frameworks for learning dynamic control. Need new software stack for emerging research in RL in computer systems:

- Deep learning libraries like TensorFlow provide low level numerical operators, neural network facilities but no RL abstraction
- Existing reinforcement learning libraries focused on simulation use cases, games, not general purpose APIs
- We propose TensorForce, an applied RL library focusing on usability. TensorForce provides a set of common algorithms and modelling techniques through a declarative interface

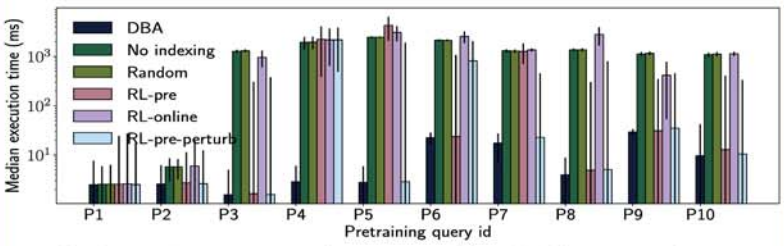


https://github.com/reinforceio/tensorforce

Results and analysis

Task: Dynamic index selection in NoSQL database. Typically performed manually/tool-assisted via profiling.

- As index creation can take seconds to minutes, training times are prohibitive for learning without any prior information
- Idea: Use example traces from existing applications to guide model creation via deep reinforcement learning from demonstration
- Pretrain from demonstrations, adapt to specific workload at runtime



Evaluate 10 query templates on workload with non-stationary popularity:

- Human DBA indices created statically once
- Different RL modes (online only, pretraining with online adaption, pretraining with adversarial perturbations)
- New benchmarks needed to learn to adapt to larger set of query structures