Reinforcement Learning for Database Indexing

Gabriel Paludo Licks
Graduate Program in Computer Science - School of Technology
Pontifical Catholic University of Rio Grande do Sul - PUCRS
Porto Alegre, Brazil
gabriel.paludo@acad.pucrs.br

Abstract

This paper describes a work proposal for using Reinforcement Learning (RL) for database indexing. While the index configuration of a database has a direct impact on its performance and response time, choosing the right index configuration is not a trivial task. We propose to implement a RL agent using the Q-Learning algorithm to explore different index configurations in a TPC-H relational database model, using its benchmark performance metric as a reward for the agent. Among all possible configurations explored by the agent, it is expected to find an optimal index configuration with a maximum performance metric obtained.

Introduction

Databases with large volumes of data have a constant challenge of achieving satisfactory response time for its users. Among many other techniques, the use of indexes in a database is one of the solutions for performance improvement, especially when it comes to processing complex queries (Ramakrishnan and Gehrke 2000). More than finding the correct columns to be indexed, it is crucial to have a balance in the amount of indexed columns. Too many indexes, for example, could result in an overhead during the index look-up process. Thus, this is a task that needs to be performed with caution, as the indexing configuration directly impacts on a database's overall performance.

That said, some authors have presented a variety of approaches for database indexing. Basu et al. proposed the use of Reinforcement Learning for suggesting which indexes to create or drop by one query at a time. A study from Sharma, Schuhknecht, and Dittrich, used a Reinforcement Learning approach to recommend a column to be indexed, given a set of queries provided by the TPC-H Benchmark. Both authors made use of Reinforcement Learning (RL) for index suggestion, being the first one limited to working with one query at a time, and the latter by suggesting only one column to be indexed given a set of queries.

Therefore, RL is a potential approach for database index tuning. This is a proposal for using a RL agent to act over a database, altering its index configuration by creating or dropping column indexes, using the Q-Learning algorithm for exploring different configurations and the TPC-H Benchmark for database workload. In addition, unlike previous work, this proposal aims to give index recommendations in between whole query workloads and, lastly, retrieve the best resulting set of indexes explored by the agent. In the next sections, we have a technical approach for this proposal and we break down the tasks that will need to be accomplished.

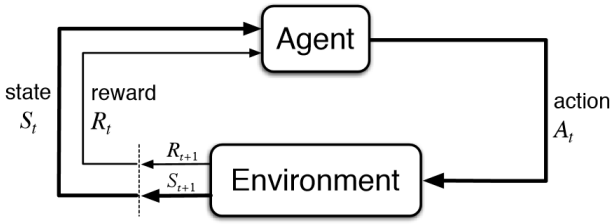

Figure 1: Agent-environment interaction in a MDP

Technical Approach

This work relies on an implementation of a Q-Learning agent, which will act over a database set of indexes to explore different configurations. Our RL problem can be formulated as a Markov Decision Process (MDP). Figure 1, from Sutton et al., illustrates the agent-environment interaction in a MDP. Thus, there are a few aspects that need to be defined: the environment in which our agent will be acting in; how the state of this environment will be represented; which actions our agent will be able to execute; and a reward measure as a feedback from the environment to the agent.

In our case, the environment is the database itself. For that, we will use the structure and relational model provided by the TPC-H Benchmark. Moreover, the chosen DBMS we will be working with is the MySQL, influenced by an implementation for benchmarking with TPC-H on off-the-shelf hardware by Thanopoulou, Carreira, and Galhardas. The queries and the data populating the database have been chosen to have broad industry-wide relevance (TPC 1998). In between every index recommendation, the whole query workload provided by the benchmark will be executed, in order to check how effective a given index configuration is.

Our environment state will be defined by the database's index configuration. That is, which columns are indexed and which are not. For that, we can easily represent it with a bitmap containing 0 if a given column is not indexed, or 1 if the column is already indexed in the database. Therefore, our available actions will also be based on this bitmap, checking whether we can create or drop an index on a given column. As a result, the amount of possible states corresponds to the number of combinations of columns indexed/not indexed in the database.

Finally, one of the tasks of RL is to use observed rewards to learn an optimal (or nearly optimal) policy for the environment (Russell and Norvig 2016), therefore our agent needs a reward function. The performance metric reported by TPC-H, called the TPC-H Composite Query-per-Hour Performance Metric (QphH@Size), reflects multiple aspects of the capability of the system to process queries (TPC 1998). In order to get a feedback whether a given index configuration has increased or decreased throughput, we will be using the TPC-H performance metric as our reward function. The highest performance metric rewarded will be used to evaluate whether the agent found the best index configuration among the explored ones.

Project Management

The macro tasks involving this project are: (1) studying and implementing the TPC-H Benchmark performance test; and (2) implementing the RL agent integrating its features with the database. However, it is likely that other micro tasks arise throughout the implementation process. A tentative time schedule for accomplishing this work, on a weekly basis, is:

- Week 1 (15/10): TPC-H Benchmark specification review, script implementation and testing
- Week 2 (22/10): Agent implementation and integration with database
- Week 3 (29/10): Agent implementation and testing different configurations
- Week 4 (05/11): Analyzing results and writing paper
- Week 5 (12/11): Writing paper and proofreading

The first macro task will be carried out by implement a Python script for running performance tests strictly following the benchmark specifications, taking as a reference the previously mentioned work done by Thanopoulou, Carreira, and Galhardas. The latter task consists of implementing agent-related classes such as Environment, State, Action, the Agent itself and a Database class responsible for interactions with the database. Moreover, there is a possibility of testing more than one exploration function for the agent and trying different values for learning rate and discount factor.

Conclusion

Creating indexes is a task that Database Administrators (DBAs) recurrently have to perform. When it comes to deeper analyzing the cost that an index implies in the database performance, it is a time consuming task to be performed frequently. Especially, when analyzing many possible recommendations of indexes for achieving the best set, the RL agent has the ability to explore a higher number of combinations. Thus, after developing the proposed tasks and executing the agent in the environment, it is expected to find an optimal index configuration with a maximum performance metric obtained among all possible configurations explored by the agent.

References

Basu, D.; Lin, Q.; Chen, W.; Vo, H. T.; Yuan, Z.; Senellart, P.; and Bressan, S. 2016. Regularized cost-model oblivious database tuning with reinforcement learning. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXVIII*. Springer. 96–132.

Ramakrishnan, R., and Gehrke, J. 2000. *Database management systems*. McGraw Hill.

Russell, S. J., and Norvig, P. 2016. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Sharma, A.; Schuhknecht, F. M.; and Dittrich, J. 2018. The case for automatic database administration using deep reinforcement learning. *arXiv preprint arXiv:1801.05643*.

Sutton, R. S.; Barto, A. G.; Bach, F.; et al. 1998. *Reinforcement learning: An introduction*. MIT press.

Thanopoulou, A.; Carreira, P.; and Galhardas, H. 2012. Benchmarking with tpc-h on off-the-shelf hardware. *14th International Conference on Enterprise Information Systems* 205–208.

TPC. 1998. Transaction performance council website (tpc). http://www.tpc.org/.