Can Transfer Learning be used to build a Query Optimizer?

Yunjia Zhang, Yannis Chronis, Jignesh M. Patel, Theodoros Rekatsinas
{yunjia,chronis,jignesh,thodrek}@cs.wisc.edu
University of Wisconsin-Madison

In this paper, we propose the problem of transferring a query optimizer. Our problem statement is: Given a database System A that has a "good" optimizer, and another database System B that has a poor or no optimizer, but has a similar set of physical operator implementations, can we design a query optimization module for System B that learns optimization patterns from System A and then use this module to optimize queries for System B? The conceptual view of this problem statement is depicted in Figure 1.

The key observation driving the above problem statement is that building query optimizers, especially for new data platforms, is hard, and traditional database systems that have "good" optimizers take years if not decades to harden. The first part of our motivation for studying this problem is purely intellectual – we are simply curious as to how far we can go with the approach stated above. The second part of our motivation is to explore if the approach above can be practically used to improve the query optimization process for a data platform that does not have a good optimizer.

A subtle but critical aspect of the problem statement above is that both systems A and B must have a similar set of core physical operators. We note that basic "good" implementations for the core relational operators are well known (e.g., hash-join, index nested-loops join, index-based selection, hash aggregation, etc.), and it is not too hard to implement these basic operators. Thus, it is reasonable to require that any modern data platform that wants to use our approach must have reasonable implementations of these basic relational operator algorithms. A nuanced but important point is that even if System A has more finely tuned operator implementations than System B, the query optimizers task is to differentiate between these individual operators in a given system. Hence, what really matters from a query optimizer's perspective is the relative performance of the algorithms for a given relational operator in a given system. In other words, there is the potential for a transfer-based query optimizer to work even if the two systems A and B have very differently tuned relational operator evaluation methods.

Future Directions. Given the potential of transferring query optimizers, we list four directions for future research study as follows.
Generalizing Learned Optimization Models. Although recent ML methods are proven to be effective in various aspects of query optimization [1, 2], a common challenge with current learning-based methods is that their knowledge is defined by their training data set, limiting their applicability when the data set is frequently updated, and/or the learning-based methods require retraining to be effective. In the scenario of transferring query optimizer, this limitation requires the data on the target System B to be migrated onto System A to train the models, which may cause large training overheads. Therefore, to accommodate the transferring application, the ML models need to be applicable or adaptive to different data sets from the training data.

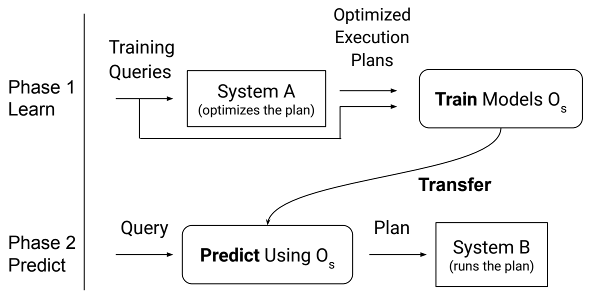

Figure 1: The phases of transferring the optimizer of System A to System B.

Characteristic-based Transferring. After decades of development, there are a decent number of well-performing query optimizers from systems like PostgreSQL, Microsoft SQL Server, etc. However, optimizers can be highly tuned specifically for their systems, so the plans generated by different optimizers can vary significantly. Therefore, future studies may focus on choosing the the most appropriate source query optimizer for a given target System B.
Adaptive Transferring. The plan generated by a transferred query optimizer may be the best plan for System A but not for System B. One reason for this difference could be the different designs of the two systems. An improved transferring solution would use the transferred optimizer as a starting point and adapt the models as more queries are executed on target System B, and their corresponding query execution costs are observed (for example [1]). An open question here is whether we can incrementally adjust the transferred ML model.
Interpretable Transferring. In the setting of query optimizer transferring, where the transferred optimizer is not inherently designed for the system where it is being used, a transferred model does not provide the same query optimization introspection utilities as a traditional optimizer. This increases the need of interpreting the decisions made by the transferred models. A direction for future work is to explore if interpretable ML methods could be used.

REFERENCES

[1] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. 2021. Bao: Making learned query optimization practical. In *Proceedings of the 2021 International Conference on Management of Data*. 1275–1288.

[2] Lucas Woltmann, Claudio Hartmann, Maik Thiele, Dirk Habich, and Wolfgang Lehner. 2019. Cardinality estimation with local deep learning models. In *Proceedings of the second international workshop on exploiting artificial intelligence techniques for data management*. 1–8.