Whoosh: Precise Data-Driven Query Planning

Maruth Goyal
The University of Texas at Austin

Aditya Akella
The University of Texas at Austin

1 Introduction

In order to efficiently execute a query on data, most database systems today perform some sort of optimzation on the logical plan, followed by a search over potential physical plans [2, 3]. However, this method is *compute-centric*. i.e., the decision stems from analysis of the computational structure of the task, in addition to coarse estimates of data properties. This is in contrast to a *data-centric* strategy [4] wherein the physical plan is picked based on fine-grained properties of the data. WHIZ [4] is a system that uses user-provided strategies to pick physical operators on-the-fly during query execution. These strategies are functions of fine-grained properties of the input data for the operator. Using this, they are able to achieve a $1.27\times$ speed-up over Apache Spark on average on the TPC-DS benchmark.

We believe there are many interesting ways to build upon this data-centric framework. In this work we consider the following:

1. **(RQ1) Code Generation:** Can we extend this framework to support code-generation?
2. **(RQ2) Strategy Optimizer:** Given a space of potential strategies, how do we pick the best one for a query?
3. **(RQ3) Learning Strategies:** Can the system self-optimize by learning better strategies over time?

2 Proposed Techniques

RQ1: Code Generation Over the past decade many systems have greatly benefitted from code-generation for query plans [1, 2, 6]. However, since WHIZ by design does not early-bind to a physical plan we cannot directly port existing techniques. Our key observation is some property π of stage σ can potentially be fully determined by some property ρ inferred at stage σ' which is a parent of σ. eg: If a strategy checks if the data is sorted, and parent stages are either `ORDER BY` or order-preserving operations, we may conclude the data is sorted. Thus we propose a procedure that given ρ applies some rules depending on the operator at σ' to infer a set of properties Σ which are true of the input data for all children stages of σ'. Observe a strategy for a node can be eagerly evaluated if all properties it depends on are propagated to the node using the defined procedure. Now, for any subplan where all nodes in this subplan have been eagerly evaluated we may utilize existing compilation techniques [5, 6].

RQ2: Strategy Optimizer Monitoring data properties is not free. Thus, strategies may depend on those most relevant to their use-case. This induces the task of finding the most relevant strategy given a dataset and query. eg: for a dataset skewed on a join-key, using a skew-aware strategy is strictly better than a skew-oblivious one. We propose a "strategy optimizer" that scores an assignment of strategies by computing the correlation of the properties monitored by a strategy, and properties that *may* be true of its input data. To do this, we maintain sets of properties $\pi_i(s)$ and $\pi_o(s)$ for each strategy s. $\pi_i(s)$ are the properties s depends on, and $\pi_o(s)$ are the properties it *may* guarantee of its output. We may in addition also consider properties that may be true by applying the propagation procedure proposed in **RQ1**. With this, we aim to optimize for (1) finding strategies most relevant to the task, and (2) assigning strategies amenable to eager evaluation as formulated in **RQ1**.

RQ3: Learning Strategies Currently, the burden of figuring out optimal strategies falls on the user. We propose developing a system that uses past queries and data to learn new strategies. In particular, we formulate this as a program synthesis task over a strategy DSL. The specification for the task comprises of a set of I/O examples. Each input example consists of a node, query, and dataset. The output is a better "transcript" of physical-plan decisions. Specifically, we consider the transcript of decisions made by WHIZ for the query, and perturb the decisions for any *one* operator to find one that's faster. The synthesized program must be a strategy for the operator of the given node that makes the better decisions for the given query and dataset.

3 Next Steps

1. We are working on prototyping the optimizer in isolation in order to test and refine its ability to pick the best strategy assignment.
2. Motivated by its code-generation engine, we are currently working on modifying Apache Spark to support strategies for on-the-fly physical planning. Our goal is to architect an efficient execution layer that supports dynamic shuffle scheduling, physical planning, code-generation, and fault-tolerance with minimal modification / integration effort for Spark.
3. For **RQ3** we must develop our own PBE engine due to issues with existing tools.

References

[1] Singlestore db code generation. https://docs.singlestore.com/db/v7.6/en/query-data/query-concepts/code-generation.html, Aug 2021.

[2] Michael Armbrust, Reynold S Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K Bradley, Xiangrui Meng, Tomer Kaftan, Michael J Franklin, Ali Ghodsi, et al. Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1383–1394, 2015.

[3] Morton M. Astrahan, Mike W. Blasgen, Donald D. Chamberlin, Kapali P. Eswaran, Jim N Gray, Patricia P. Griffiths, W Frank King, Raymond A. Lorie, Paul R. McJones, James W. Mehl, et al. System r: Relational approach to database management. *ACM Transactions on Database Systems (TODS)*, 1(2):97–137, 1976.

[4] Robert Grandl, Arjun Singhvi, Raajay Viswanathan, and Aditya Akella. Whiz: Data-driven analytics execution. In *18th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 21)*, pages 407–423, 2021.

[5] André Kohn, Viktor Leis, and Thomas Neumann. Adaptive execution of compiled queries. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 197–208. IEEE, 2018.

[6] Thomas Neumann. Efficiently compiling efficient query plans for modern hardware. *Proceedings of the VLDB Endowment*, 4(9):539–550, 2011.