CS598: Project Proposal

Yangchen Ye
University of Illinois Urbana Champaign
yy49@illinois.edu

Xixiang Liu
University of Illinois Urbana Champaign
xixiang3@illinois.edu

Zihan Shan
University of Illinois Urbana Champaign
zshan2@illinois.edu

ABSTRACT

In modern SQL execution, predicate is often pushed into table scan to leverage the metadata and indexes provided by the underlying storage format, e.g., apache parquet, for data skipping and efficient processing, and the key to success is for the storage format to provide good enough data structures to support this use case. The effectiveness of techniques employed by apache parquet, i.e., zone maps and column indexes, are limited by data partitioning strategies and predicate selectivity, whereas new techniques which claim to have robust performance boost like column sketches are only evaluated with in-memory processing and lack integration into an actual storage format like parquet. In this project, we propose a reproduction and expansion of the column sketch technique on top of apache parquet, which brings the robust performance boost to parquet and explore its effectiveness in an end-to-end benchmark with different realistic dataset distributions. We implement the column sketch algorithm as an open-source tool with support for accepting a parquet file and output modified parquet file containing the column sketch data structrues. We also build a custom dataset generation tool to support generating parquet tables with statistic properties mimicing realistic workload data, and we use this tool to explore column sketch's effectiveness on an industry data format and realistic dataset. The work demonstrates the prospect of enhancing an industry columnar storage format with new auxilary structures to make it better at the widely used SQL workflow of table scanning with predicate evaluation.

1 INTRODUCTION

Disk I/O has always been a dominant factor in SQL query execution time, and pushing predicate evaluation down into the storage has long been employed for the benefit of skipping unnecessary I/Os. In traditional transactional database systems, the SQL optimizer can transform a sequential scan followed by a filter operator to a B+ tree index scan if possible. Creating the right index can often bring 10x speedups to some queries. The basic principle still governs modern OLAP workload which operates on petabyte-scale of data. The key to the efficient execution of queries is to have good data structrues on the underlying storage to allow skipping disk I/Os.

However, existing approaches all have limitations in the context of OLAP workload[3]. The limitation of using indexes is that it only reduces I/O when the predicate has extremely low selectivity, which is often the case in transactional workload but not that much in modern analytical workload. Therefore, data format used by modern OLAP database systems like apache parquet[1] often use a more lightweight data structure called zone map. The idea is to horizontally divide the table into many partitions, and keep metadata about each single partition like the min, max of each column values in that partition. The benefit of zone map comes from the ability to skip entire partitions if we can determine predicate on metadata alone. Unfortunately, it only works well if the partitions are clustered on the right column, so that the min and max values are clustered. On the other hand, if the column is uniformly distributed across partitions, then there is a high chance that each partition's min and max resembles global min and max and we won't be able to skip any partitions at all.

Column sketch, on the other hand, is a technique that can bring robust performance increase regardless of how data is partitioned or the selectivity of a given query[2]. It proposes a promising complement of existing techniques to make the storage format even better. In a nutshell, column sketch creates an order-preserving compression map and a compressed column over the original column and use the compressed column for query evaluation to reduce per-record memory footprint. However, the evaluation and experiment for the original project is limited to in-memory datasets rather than the whole pipeline of reading from storage, and it didn't propose possible integrations with widely used storage format like parquet. For example, the paper doesn't discuss how the compression map could be serialized to persistent storage and whether the performance boost is still robust when we read both the column and the compression map metadata from disk.

This project is aimed at reproducing and expanding the original column sketch project. We implement the algorithm as an open-source tool and design serialization and deserialization strategies for persisting the data structrues to parquet files. There are two main questions we answered through our evaluation of the project. First, we show that column sketch is effective in a complete parquet scanning benchmark, i.e., the overhead of reading additional structures from storage doesn't outweight its performance gain. Second, we demonstrate that the storage overhead of column sketch is reasonable and it is feasible to apply it to real world tables of large size.

Through this project, we wish to demonstrate the prospect of improving the modern columnar storage format by bringing in new data structrues to support efficient scanning with predicate in a robust fashion.

REFERENCES

[1] [n.d.]. Apache Parquet. https://parquet.apache.org/
[2] Brian Hentschel, Michael S. Kester, and Stratos Idreos. 2018. Column Sketches: A Scan Accelerator for Rapid and Robust Predicate Evaluation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.*
[3] Xinyu Zeng, Yulong Hui, and et al. 2023. An Empirical Evaluation of Columnar Storage Formats. *arXiv preprint arXiv:2304.05028* (2023).