CS511 SP24 Final Project Update 3

Author: Yangchen Ye, Xixiang Liu, Zihan Shan

This week's vector

Now that we've got a minimal viable product of column sketch, next week's vector would be to come up with reasonable first-step integration with apache parquet. Specifically we will try out several serialization/deserialization design of the column sketch data structure and compressed code. Next week's focus would be to come up with a reasonable serialization plan and study the storage overhead.

This week's plan

We will start with a 'hello-world' style prototype. We will implement a binary which accepts a parquet table, apply column sketch on the column and output a new parquet table with extra columns which are compressed code. We will also emit a separate binary file containing the serialized compression map for the columns. We will study the effectiveness of this scheme, and if it works well, we could implement better integration by merging the data structure file into custom parquet pages.

This week's result

After some trial and error, we implemented the following serialization scheme: We will leave columns in the original table untouched, and add a new column called "${column_to_be_sketched}.sketch", which is all UINT8 code. Then we will serialize the column sketch data structure as a stream of three structures: a histogram as an array of endpoint values and a bitvector indicating uniqueness at each entry. The column sketch structure would be | data_type | histogram (256 * sizeof(data_type)) | bitvector (256) |.

We currently only support running column sketch on one column, so there's no need for extra metadata for tracking the mapping between compression maps and columns. We found that due to the nice property of UINT8 compressed code, the storage overhead is acceptable for medium sized singe floating point tables we tested, because parquet is usually able to come up with reasonable encoding for the compressed column.

Next week's vector

Next week's task would be to run controlled end-to-end experiment on the performance of data scanning and predicate evaluation on augmented parquet vs. real parquet. We expect the damage of a sketch miss will be amplified in a disk-involved environment as disk read is always done in chunks rather than a single value. The vector we would like to test is if the win is still salient when put in a realistic disk workload.

Next week's plan

We would implement controlled experiment with controlled dataset generated from our own tools, which can have custom level of sortedness and distribution. We want to reproduce the strength of zone-map approach for clustered columns (we use sortedness as a proxy for clusteredness) and column sketch's superiority for more or less randomly distributed columns.

One sentence per person

Xixiang Liu: I further learn the rationale behind the codes, and I am working on extending the basic version parquet converter.

Zihan Shan: I am working on debugging and adding more features to the dataset generator, in order to satisfy our testbench requirements.

Yangchen Ye: I coded the initial version of the transformer which reads and runs column sketch to augment the parquet file.