The Deconstructed Warehouse: An Ephemeral Query Engine Design for Apache Iceberg

Ryan Curtin
Independent Researcher
ryan@ratml.org

Jacopo Tagliabue
Bauplan Labs
jacopo.tagliabue@bauplanlabs.com

ABSTRACT

The rise of open formats (e.g. Apache Iceberg, Delta Lake) and single-node vectorized engines (e.g. DuckDB, DataFusion) have been important recent trends in data systems. Perhaps surprisingly, these trends did not interact to produce a credible, cloud-first OLAP experience *yet*: while open formats are a staple of large scale lakehouses [5], and proprietary offerings emerged over the DuckDB format [1], managed Iceberg-native experience are still lacking.

In the spirit of the 'Composable Data Manifesto" [2], we show how to achieve warehouse-like capabilities (with very modest resources) through ephemeral functions and disaggregated components—storage, data catalog, planning, caching, execution. *We summarize our contributions as follows*:

(1) we present a novel design for integrating data catalogs, open formats and single-node engines into a "deconstructed warehouse";
(2) we motivate and detail a new command we implemented in our DuckDB fork, `EXPLAIN SCANS`, which sits in between the logical and the physical plan as an intermediate optimization;
(3) we present preliminary results when benchmarking our I/O approach against sensible alternatives.

Fig. 1 illustrates the three main components of the architecture: a multi-tenant parser-planner extracts from queries the relevant scan operations, with optimized columnar and predicate push-downs (the `EXPLAIN SCANS` API below); an Iceberg catalog maps table scans to byte-range operations over object storage; a purpose-built [3] FaaS runtime provides ephemeral compute capacity to run an embedded engine on the scans. It's important to note that as long as the parser-planner and the engine speaks the same SQL dialect, the architecture is composable.

Fig. 2 represents an execution at the runtime level. A user request is decomposed in three ephemeral functions, all communicating through Arrow with no serialization overhead: a scan function, which reads from object storage (or a cache) and returns the corresponding scans; the single-node query engine, returning the result tuples after running the query on the scans; an ephemeral Flight server, streaming the results to the user. Mapping queries to S3 scans required us to add a command—`EXPLAIN SCANS`—to our planner, forked from *DuckDB*. The command decomposes a query into optimized scans: scans are then mapped to Parquet reads through the catalog. Importantly, even at this layer we maintain full modularity, allowing different engine or even data formats to be plugged with minimal changes if necessary.

On top of modularity, a second motivation for decoupling planning, S3 scans and query execution is performance. We share our preliminary benchmarks on alternative ways to run queries on an embedded FaaS engine: using a wildcard over the bucket, replacing tables with `read_parquet` over the files returned by the Iceberg Scan API, or relying on an Iceberg plugin.[1] While separating reads from query execution in theory leaves performance on the table (as we can't interleave I/O and compute), we are able to compensate by reading more effectively from object storage than DuckDB built-in methods, as well as performing aggressive I/O optimizations [4] while keeping execution stateless.

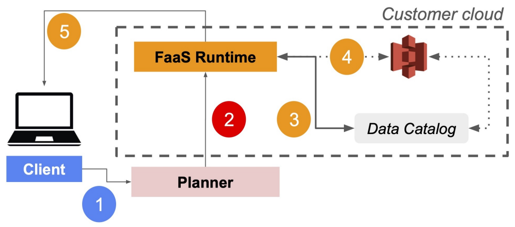

Figure 1: A FaaS-based architecture with storage-compute decoupling, ephemeral serverless compute and modular planning.

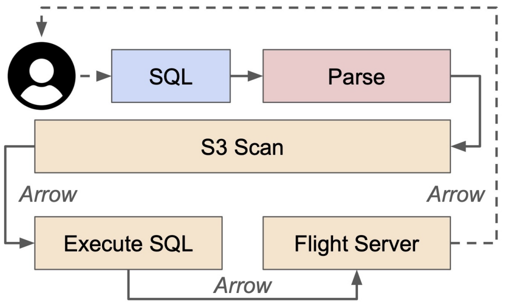

Figure 2: Query execution as a sequence of functions exchanging Arrow streams.

REFERENCES

[1] RJ Atwal, Peter A Boncz, Ryan Boyd, Antony Courtney, Till Döhmen, Florian Gerlinghoff, Jeff Huang, Joseph Hwang, Raphael Hyde, Elena Felder, et al. 2024. MotherDuck: DuckDB in the cloud and in the client.. In *CIDR*.
[2] Pedro Pedreira, Orri Erling, Konstantinos Karanasos, Scott Schneider, Wes McKinney, Satya R Valluri, Mohamed Zait, and Jacques Nadeau. 2023. The Composable Data Management System Manifesto. *PVLDB* 16, 10 (June 2023), 2679–2685.
[3] Jacopo Tagliabue, Tyler Caraza-Harter, and Ciro Greco. 2024. Bauplan: Zero-copy, Scale-up FaaS for Data Pipelines. In *WoSC*.
[4] Jacopo Tagliabue, Ryan Curtin, and Ciro Greco. 2024. FaaS and Furious: abstractions and differential caching for efficient data pre-processing. In *2024 IEEE International Conference on Big Data (BigData)*. 3562–3567. https://doi.org/10.1109/BigData62323.2024.10825377
[5] Matei A. Zaharia, Ali Ghodsi, Reynold Xin, and Michael Armbrust. 2021. Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics. In *Conference on Innovative Data Systems Research.*

[1] https://duckdb.org/docs/extensions/iceberg.html

Proceedings of the VLDB Endowment. ISSN 2150-8097.