Improving SQL Query Execution of Distributed Query Engines on Object-Based Computational Storage through Multi-Layered Offloading

Soon Hwang[1], Junhyeok Park[1], Junghyun Ryu[1], Jungahn Park[2], Jeongjin Lee[2], Jungki Noh[2]
Soonyeal Yang[2], Woosuk Chung[2], and Youngjae Kim[1]
[1]Sogang University, Seoul, Republic of Korea, [2]Memory System Research, SK hynix Inc.

Background. In modern distributed SQL query engines for data analytics, such as Presto [4], a common challenge arises when compute nodes must retrieve large datasets from storage nodes, incurring unnecessary data movement costs, even though only a small portion of the data is relevant to the actual query [7]. S3 SELECT [1] attempts to mitigate this issue by allowing certain filter operations, such as WHERE clauses, to be executed on the storage side. While this reduces data transfer, it is limited to supporting only simple queries composed of SELECT and WHERE clauses.

Recently, Los Alamos National Laboratory (LANL) and SK hynix introduced the Object-based Computational Storage (OCS) [5], offering a more versatile approach to these challenges. The OCS system consists of OCS Front-End (OCSFE) servers that serve as gateways with S3 compatibility using Versity [6] for multiple OCS Array (OCSA) servers. Each OCSA server contains multiple storage devices and offers object storage capabilities. Unlike S3 SELECT, which is limited to specific query types, the OCS system supports offloading platform-independent query plans via the Substrait [3], regardless of the type of query. OCS also provides broad execution engine selections (e.g., DuckDB [2]) with customization availability and Substrait-compatibility, while S3 SELECT only supports pushdown execution with internal service. This enables more general query execution, not limited to SELECT operations.

Design and Challenges of Multi-Layered Offloading for SQL Queries in Object-Based Computational Storage. The current OCS system shows promise in addressing data analytics challenges, but it still lacks certain key functionalities to fully integrate with existing analytics platforms like Presto. One significant limitation is the absence of a connector or plugin that enables offloading query execution to the OCS. This shortcoming poses a challenge for widespread adoption of OCS in real-world environments. Furthermore, the current OCS architecture forces all offloaded query plans to be executed at the OCSA layer, leaving the OCSFE, which has substantial compute capabilities, underutilized. This single-layer approach results in inefficiencies and missed opportunities to balance the computational load across the OCS system.

To overcome this, we first configured the OCSFE as an additional execution layer, deploying DuckDB to execute query plans. We then developed the OCS-Decomposer, a module that decomposes offloaded query plans into sub-plans at the operator level. These sub-plans can be executed across both the OCSFE and OCSA, with intermediate results transferred between them. This multi-layered vertical query execution is expected to improve resource utilization and query latency by distributing the workload more evenly across the system. In parallel, we are also working on integration with Presto, enabling Presto to translate user query into Substrait-formatted query plans and send it to OCS. Figure 1 shows the architecture of the Presto-integrated OCS system that enables a multi-layered vertical query offloading.

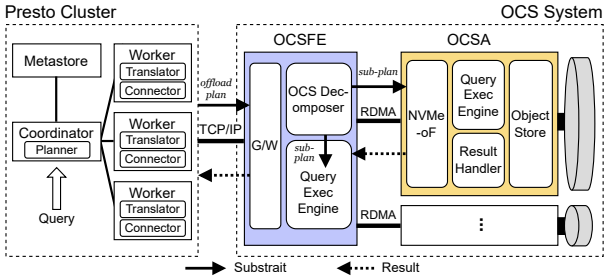

Figure 1: Pushdown Framework for Multi-Layered Vertical Query Execution in the Presto-Integrated OCS System.

However, beyond addressing aforementioned technical challenges, a larger issue remains: the need for an heuristic algorithm that can dynamically coordinate query execution between OCS and analytics platforms. We are currently aiming to achieve three objectives. **First**, the algorithm should be able to decompose query plans and assign sub-plans to the appropriate layer for offloading. Since analytics platforms view the OCS system as a blackbox, the platform only decides whether to offload a query based on its selectivity. Once offloaded, the OCS system decomposes the query plan and assigns sub-plans to each layer, depending on data movement. **Second**, the algorithm should consider the resources available at each layer. The analytics platform can estimate the OCS system's status by checking the number of pending query requests. Once the query is offloaded to the OCS system, the system can monitor its components by tracking the requests currently being handled. **Third**, since the goal of OCS is to create a platform-independent computational system, the OCS system should operate with an algorithm that can be seamlessly integrated into multiple platforms. To achieve this, we are currently developing algorithm logic that requires minimal changes to the analytics platform while providing OCS system features through a plugin or connector. Here are some research questions we are considering for expanding the algorithm.

- ***RQ1.*** What can be the factors for deciding the decomposing position (operator) for multi-layered vertical SQL query execution?
- ***RQ2.*** How can we elastically orchestrate query execution in case of storage system scaling?
- ***RQ3.*** What should be considered for multi-layered computational storage system to be operated regardless of client side platforms?

REFERENCES

[1] AWS. 2024. Using S3 Select Pushdown with Presto to improve performance. https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-presto-s3select.html.
[2] DuckDB Foundation. 2024. DuckDB. https://duckdb.org/.
[3] Substrait Project. 2024. Substrait. https://substrait.io/.
[4] Raghav Sethi, Martin Traverso, Dain Sundstrom, David Phillips, Wenlei Xie, Yutian Sun, Nezih Yegitbasi, Haozhun Jin, Eric Hwang, Nileema Shingte, and Christopher Berner. 2019. Presto: SQL on Everything. In *Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE).*
[5] SK hynix. 2024. Toward Open Standardized Object-Based Computational Storage for Big Data Analytics. Presented at FMS 2024.
[6] Versity. 2024. Versity Gateway. https://www.versity.com/products/versitygw/.
[7] Qing Zheng. 2022. Kinetic Campaign: Speeding Scientific Data Analytics with Computational Storage Drives. Presented at SDC 2022.