Toward Standardized, Open Object-Based Computational Storage For Large-Scale Scientific Data Analytics

Qing Zheng
Los Alamos National Laboratory
Los Alamos, NM, USA
qzheng@lanl.gov

Jason Lee
Los Alamos National Laboratory
Los Alamos, NM, USA
jasonlee@lanl.gov

Dominic Manno
Los Alamos National Laboratory
Los Alamos, NM, USA
dmanno@lanl.gov

Gary Grider
Los Alamos National Laboratory
Los Alamos, NM, USA
ggrider@lanl.gov

CCS CONCEPTS

• **Information systems** → **Storage architectures**; *Online analytical processing engines*; • **Hardware** → *External storage.*

EXTENDED ABSTRACT

With the increasing reliance of scientific discovery on insights derived from massive datasets, attaining fast, efficient data analytics performance has become a key component of modern HPC data center engineering. While fast interconnection network, fast storage media, and a proliferation of open-source data analytics software such as Apache Hive [2], Presto [7], and DuckDB [5] have made the deployment of distributed, high-performance data analytics platforms much more feasible and convenient than it was many years before, bottlenecks still exist that prevent applications from reaching a higher level of performance.

Most notably, excessive data movement, being one of these performance barriers, occurs when queries of very high data selectivity continue to be processed in a form that resembles a full dataset scan. That is, despite that a query may be interested in only a tiny subset of rows or columns in a large dataset, it is still processed by reading all of that dataset from storage nodes to worker nodes.

This approach results in a majority of rows or columns being transferred unnecessarily, as they will eventually be discarded by the worker nodes, with only those that match the query criteria being returned to the querying program. Unnecessary data transfer can lead to substantial delays in query response times, as data movement remains a time-consuming process overall. In addition, reading an entire dataset back into worker nodes may require the same amount of memory as used by the original simulation, which can be petabytes. Once the data is loaded, the subset of data chosen for analysis can be many orders of magnitude smaller than the original dataset, making the full data load and the memory allocated for it an inefficient use of supercomputer resources.

Building indexes alongside data helps. But indexing data itself might incur massive data movement and may take a long time to complete. Write-time data partitioning and indexing methods allow for reduced data movement during index construction, but a significant amount of compute node resources might have to be dedicated to the process and partitioning data typically only accelerates queries on a single dimension — multi-dimensional queries remain unaccelerated. Computational storage [1, 3, 6], with its ability to perform computation very close to data, provides new ways of accelerating queries by permitting direct query processing within storage. This enables transferring only data relevant to a query to a querying client rather than the entire dataset.

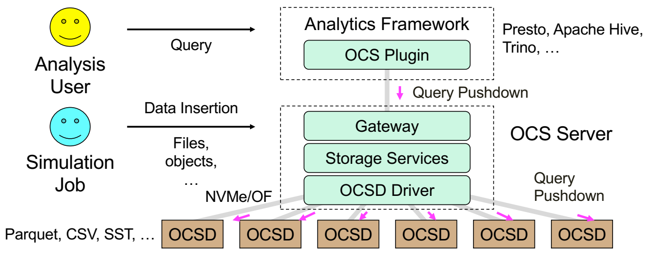

Figure 1: Object-based Computational Storage enabling analytics offload under popular storage services storing popular data formats.

This research is to establish and demonstrate an open, object-based computational storage architecture to drive minimized data movement and more efficient data retrieval performance for queries against large scientific datasets.

Open standards facilitate interoperability, community support, and vendor neutrality. Just as NFS sets the protocol for network attached storage and ANSI T10 [4] defines SCSI devices' Object-based Storage Device (OSD) command set, we advocate for a similar standardization effort for object-based computational storage. **We envision a standard, high-level interface for storage server level query pushdown and a standard, low-level NVMe command set for device-level query offload.** We call the high-level interface Object-based Computational Storage (**OCS**) interface and the low-level interface Object-based Computational Storage Device (**OCSD**) interface. As Figure 1 shows, a typical OCS analytics stack consists of an OCS-aware analytics framework, storage servers supporting the high-level interface, and NVMe computational storage devices implementing the low-level command set.

Our work is motivated by **1)** the growing interest in object storage as a major data analytics data source, **2)** a lack of open standards for delegating query plans to object-based storage to speed up selective data retrieval, and **3)** the advent of NVMe as a modern replacement for the outdated SCSI storage device interface upon which the old OSD command set was built.

Collaboration has already taken place with industry partners including SK hynix, AirMettle, NeuroBlade, and Versity. Deliverables include an open-source reference implementation for a gateway server implementing the high-level OCS interface, an open-source reference implementation for an underlying storage device implementing the low-level OCSD command set, demonstration of interoperability of proprietary products from SK hynix, AirMettle, and NeuroBlade, and test vehicles acting as servers or clients of both the high-level and the low-level interface.

This manuscript has been approved for unlimited release and has been assigned LA-UR-23-30351.

REFERENCES

[1] Anurag Acharya, Mustafa Uysal, and Joel Saltz. 1998. Active Disks: Programming Model, Algorithms and Evaluation. In *Proceedings of the Eighth International Conference on Architectural Support for Programming Languages and Operating Systems*. https://doi.org/10.1145/291069.291026

[2] Jesús Camacho-Rodríguez, Ashutosh Chauhan, Alan Gates, Eugene Koifman, Owen O'Malley, Vineet Garg, Zoltan Haindrich, Sergey Shelukhin, Prasanth Jayachandran, Siddharth Seth, Deepak Jaiswal, Slim Bouguerra, Nishant Bangarwa, Sankar Hariappan, Anishek Agarwal, Jason Dere, Daniel Dai, Thejas Nair, Nita Dembla, Gopal Vijayaraghavan, and Günther Hagleitner. 2019. Apache Hive: From MapReduce to Enterprise-Grade Big Data Warehousing. In *Proceedings of the 2019 International Conference on Management of Data*. https://doi.org/10.1145/3299869.3314045

[3] Kimberly Keeton, David A. Patterson, and Joseph M. Hellerstein. 1998. A Case for Intelligent Disks (IDISKs). *SIGMOD Rec.* (1998). https://doi.org/10.1145/290593.290602

[4] D. Nagle, M. E. Factor, S. Iren, D. Naor, E. Riedel, O. Rodeh, and J. Satran. 2008. The ANSI T10 object-based storage standard and current implementations. *IBM Journal of Research and Development* (2008). https://doi.org/10.1147/rd.524.0401

[5] Mark Raasveldt and Hannes Mühleisen. 2019. DuckDB: An Embeddable Analytical Database. In *Proceedings of the 2019 International Conference on Management of Data*. https://doi.org/10.1145/3299869.3320212

[6] Erik Riedel, Garth A. Gibson, and Christos Faloutsos. 1998. Active Storage for Large-Scale Data Mining and Multimedia. In *Proceedings of the 24rd International Conference on Very Large Data Bases.*

[7] 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 *2019 IEEE 35th International Conference on Data Engineering*. https://doi.org/10.1109/ICDE.2019.00196