Using Ceph's BlueStore as Object Storage in HPC Storage Framework

Kira Duwe
kira.duwe@ovgu.de
Otto von Guericke University Magdeburg
Magdeburg, Germany

Michael Kuhn
michael.kuhn@ovgu.de
Otto von Guericke University Magdeburg
Magdeburg, Germany

Abstract

In times of ever-increasing data sizes, data management and insightful analysis are amidst the most severe challenges of high-performance computing. While high-level libraries such as NetCDF, HDF5, and ADIOS2, as well as the associated self-describing data formats, offer convenient interfaces to complex data sets, they were built on outdated assumptions of storage systems and interfaces. They mostly rely on the POSIX interface that researchers have been aiming to replace for decades. Among others, its strict file semantics are not suitable for current HPC systems. As object storage has become increasingly prominent to store datasets of data formats like HDF5, providing a scalable object store backend is necessary. Therefore, we looked into Ceph's object store BlueStore and developed a backend for the storage framework JULEA that uses BlueStore without the need for a full-fledged working Ceph cluster. This way, we significantly reduce the prerequisites of running it on an existing HPC cluster. BlueStore works directly on a raw block device and thereby circumvents the problems of other Ceph storage backends like FileStore and KStore.

In a first evaluation, we examine the performance of BlueStore and compare it to a POSIX-based solution which shows our prototype is functional yet not optimized enough to keep up with the POSIX-based object store. For example, the peak for explicitly synced writes is 50 MB/s for POSIX with a block size of 4,096 kiB and thereby twice as high as BlueStore's with 20.5 MB/s.

CCS Concepts: • **Software and its engineering → File systems management**.

Keywords: filesystem, object store, BlueStore, JULEA, POSIX, storage framework, data management

CHEOPS '21, April 26, 2021, Online, United Kingdom

ACM ISBN 978-1-4503-8302-8/21/04.
https://doi.org/10.1145/3439839.3458734

ACM Reference Format:
Kira Duwe and Michael Kuhn. 2021. Using Ceph's BlueStore as Object Storage in HPC Storage Framework. In *Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems (CHEOPS '21), April 26, 2021, Online, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3439839.3458734

1 Introduction

Scientific research has become increasingly data-intensive. High-resolution simulations generate rapidly growing data sets that require complex management solutions to perform meaningful analysis. To these management solutions, efficient data sifting techniques are essential.

However, the hardware and software architecture of large-scale systems in high-performance computing (HPC) complicate this task. In order to satisfy the demand for high capacity and high velocity at the same time, typically a hierarchy of different storage hardware is used.

Another factor increasing the complexity and thus complicating data retrieval is the software stack on top. A typical I/O (input/output) stack looks similar to Figure 1(a). The separation between the individual layers allows exchanging them relatively easily. However, this convenience comes at the price of performance and management issues, as the layers are often optimized for different goals.

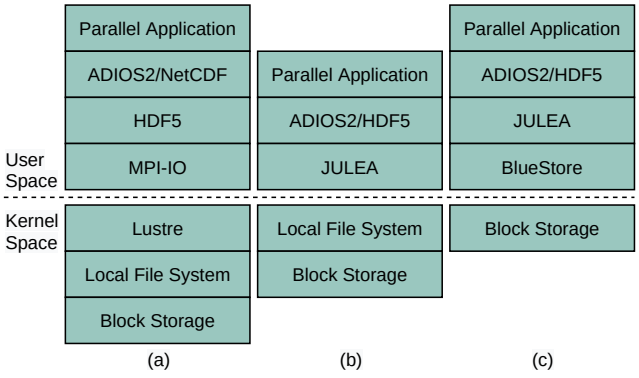


Figure 1. Exemplary HPC I/O stack with I/O libraries, self-describing data formats, I/O middleware and parallel file system for the current state (left) and when using JULEA (middle) and when using JULEA with Ceph's BlueStore (right).

I/O libraries such as NetCDF (Network Common Data Form), HDF5 (Hierarchical Data Format), and ADIOS (Adaptable IO System) are used to make data handling easier for the application developers and users. Unfortunately, the associated self-describing data formats are storage layer agnostic, capping the potential performance. Furthermore, these libraries and most parallel file systems are built on outdated assumptions and mostly rely on the POSIX (Portable Operating System Interface) I/O interface that researchers have been aiming to replace for decades. Among others, POSIX enforces strict semantics but also treats file data as an opaque byte stream, which makes it impossible to utilize structural information on lower layers.

Object Storage. A common approach to circumvent these restrictions is to store structured information like the file system metadata in key-value stores, while unstructured data, like the file content, is stored in object stores. Also, we have shown in previous work that further splitting of self-describing file formats allows coupling the data more closely with the storage system and thereby making use of the data's structural information [7]. Therefore, highly scalable key-value and object stores become increasingly relevant. There are hundreds of key-value stores and NoSQL databases.[1] Furthermore, there exist several object store solutions, especially for cloud storage, like Amazon S3 and Google Cloud Storage or open-source variants such as OpenStack Swift[2] and MinIO[3].

Another prominent example is the Reliable Autonomic Distributed Object Store (RADOS), which is at the core of Ceph and provides three services. These are object storage similar to Amazon S3 through the RADOS Gateway, a virtual block device through the RADOS Block Device, and the distributed file system CephFS built on POSIX. However, the Ceph developers turned away from building storage backends on local file systems, as they deem them unfit for distributed storage backends [3]. Instead, they designed a new object store called BlueStore that works directly on raw storage devices and runs in user-space [2]. This decision gives more control over the I/O stack and allows them to decrease performance variability.

BlueStore with JULEA. As this is a very interesting concept, we wanted to know whether it would be possible to use BlueStore without running a full-fledged Ceph Cluster. As changes in storage systems that require application changes are rarely used, we decided to integrate BlueStore into the JULEA storage framework by Kuhn [6] as it allows running HDF5 and ADIOS2 applications on top. Furthermore, JULEA is highly configurable and very modular, reducing our implementation efforts considerably.

[1]The list of NoSQL database management systems (DBMS) https://hostingdata.co.uk/nosql-database/ currently contains 225 DBMS.
[2]https://opendev.org/openstack
[3]https://github.com/minio/minio

Figure 1(b) and Figure 1(c) depict the I/O stack using JULEA and using JULEA with BlueStore. As can be seen, the stack has become simpler by moving more layers into the user-space. By using BlueStore, only the block storage remains in kernel-space.

Contribution. The main contributions of our work are listed below.

- The option to use BlueStore without running a full-fledged Ceph cluster
- A prototype for an open-source BlueStore library and BlueStore backend for JULEA[4]
- A first evaluation of the BlueStore backend in comparison to the POSIX-based object store

The paper is structured as follows. We give a short overview of the background of the Ceph storage backends as well as related work in Section 2. Afterwards in Section 3, we present the design and implementation of the JULEA backend for BlueStore. Then we evaluate this backend in comparison to JULEA's POSIX-based object store in Section 4 and give an outlook for future work in Section 5.

2 Background and Related Work

In the following, we will shortly give an overview of the Ceph architecture as well as related work.

2.1 Ceph

Ceph provides a multitude of storage backends, three of which are shown in Figure 2, highlighting their internal architecture differences [8]. Figure 2(a) depicts FileStore which is an object store built on top of POSIX where objects are mapped to files. KStore in Figure 2(b) is the evolved version without a separate Ceph journal where objects as well as metadata is handled by a key-value interface. As mentioned before, BlueStore is very different in that it gives up the local file systems entirely as shown in Figure 2(c). Figure 2(d) shows our approach using both JULEA and BlueStore. The application typically uses JULEA through our HDF5 VOL plugin or our ADIOS2 engine to avoid application changes at all costs.

BlueStore. BlueStore was designed and implemented in 2015 and has been the default storage backend for production systems of Ceph since 2017 [2]. One of the main motivations to replace local file systems completely is their lack of transactions and scalable metadata operations. Transactions in distributed file systems are typically emulated through POSIX and a write-ahead log (WAL), as can be seen in Figure 2(a).

BlueStore, however, writes the data to the raw storage device using direct I/O. The metadata is managed by RocksDB, which runs on a thin user-space file system called BlueFS [1, 2]. Furthermore, low-level file system metadata is stored in key-value stores such as extent bitmaps. Their reference

[4]https://github.com/Bella42/julea/tree/objectstore

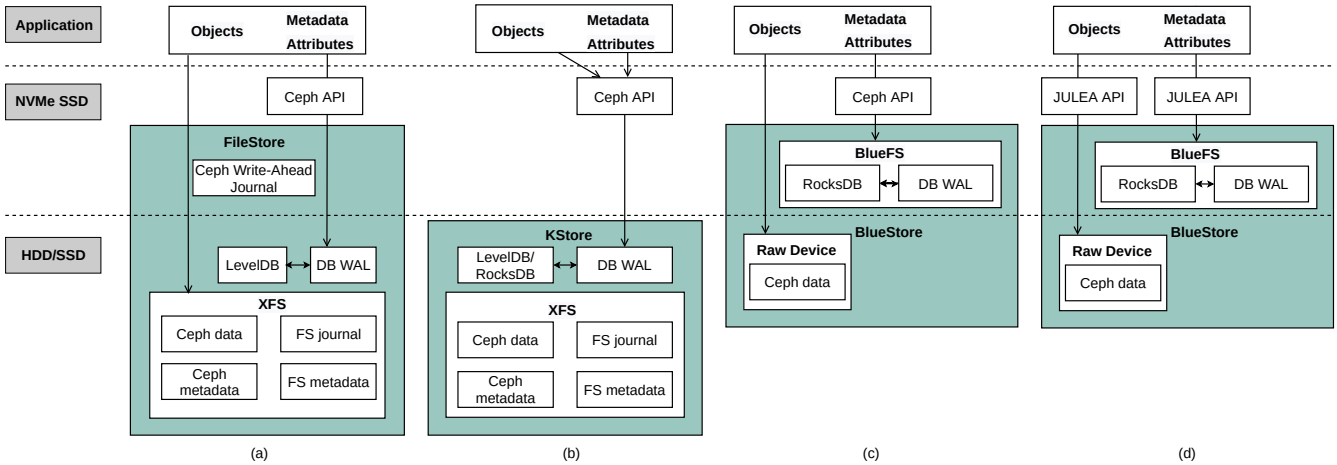


Figure 2. Architecture of the Ceph storage backends FileStore, KStore, BlueStore and BlueStore using JULEA. The figure is based on the work by Lee et al. [8].

counting and the clone operation have also been optimized. Lastly, BlueStore makes use of a space allocator that uses a fixed memory size per TB of disk space. There are also plans for support of host-managed SMR (Shingled Magnetic Recording) in BlueStore using a new zone interface. Therefore, RocksDB and thereby LSM-Trees need to be adapted to run on SMR drives [3]. Further work to support SMRs has been done by Aghayev et al. [1].

2.2 Related Work

As discussed before, object-based storage systems are efficient and scalable to overcome the limitations of the I/O stack. A prominent example is the Distributed Asynchronous Object Storage (DAOS) which aims at systems using Storage Class Memory (SCM) and NVMe storage in user space to support structured, semi-structured and unstructured data models [10]. Object-centric storage systems on HSM have also been shown to improve the performance of HDF5 when its datasets are stored in the object stores [13, 14]. Furthermore, by using Proactive Data Containers (PDC) as a tuning technique, the performance can be up to 47 times better than a highly optimized HDF5 implementation [16].

Another recent approach to optimize the mapping of datasets to object storage shows that pushing the I/O accesses to object layer allows distributing it over many servers thereby exploiting the system's parallelism[4]. Chu et al. demonstrate two possible ways using HDF5 VOL plugins on the one hand and the SkyhookDM Ceph plugin on the other hand to reorganize the data objects.

Despite the findings of the Ceph developers, Lillaney et al. argue that dual access over file system semantics and an object store API is required to satisfy all current demands [11, 12]. Another approach following the dual access concept is DelveFS which is a user space object store file system [17]. It uses custom semantics to create unique views onto the object store and is aimed at large systems using even billions of objects. Strategies proposed to solve performance problems of using file systems as storage backends were implemented in SwimStore (Shadowing with Immutable Metadata Store) in Ceph [9]. Further work on highly-scalable object storage has also been performed by ECMWF. In their system MARS [5] they use the Fields Database (FDB), which is both a library and a storage service working on Lustre as of now [15].

3 Design and Implementation

As we wanted to test whether it would be possible to use the BlueStore without a Ceph cluster, our aim in the design and implementation was to build a proof of concept prototype. Therefore, it is currently not very optimized and leaves a lot of untapped potentials.

JULEA. JULEA is a storage framework offering high flexibility. Also, it is a user-space system and thereby allows easy development, debugging as well as deployment on larger HPC clusters as no root access is necessary to run it. JULEA supports a multitude of technologies, among others SQLite, MariaDB, MongoDB, LevelDB, LMDB.

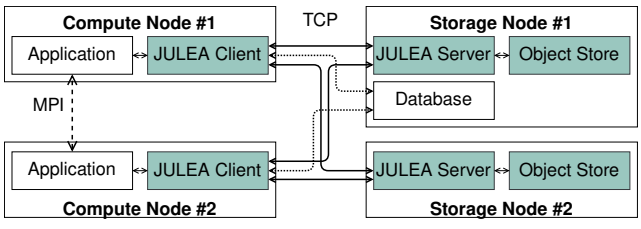
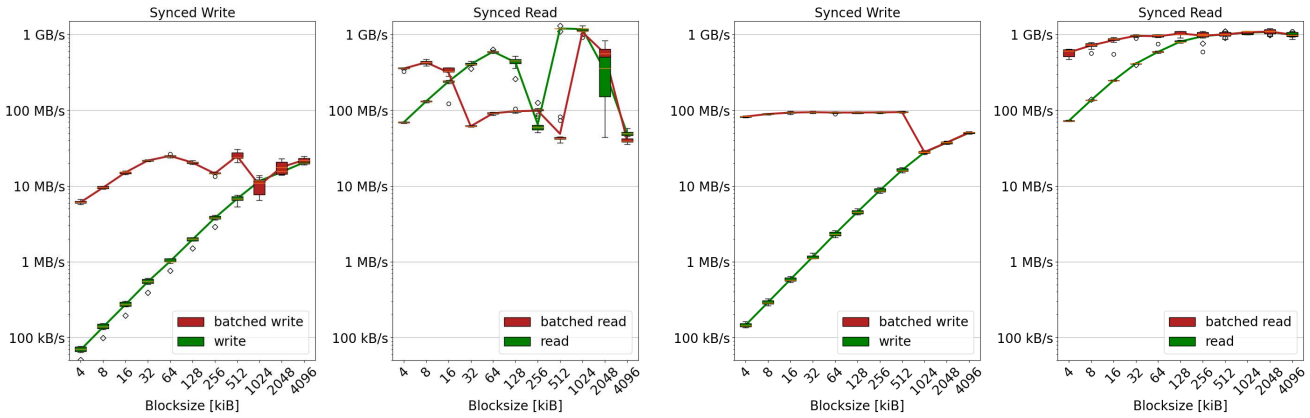


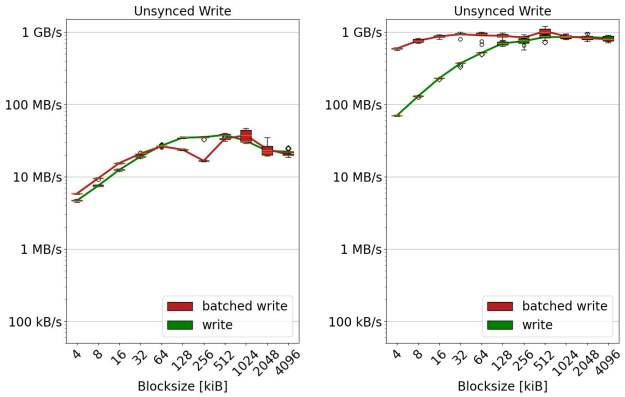
Figure 3. JULEA architecture with two distinct compute and storage nodes each.



(a) BlueStore backend with syncing

(b) POSIX backend with syncing

Figure 4. Throughput for writing and reading with explicit syncing for BlueStore and POSIX. All operations are evaluated with and without batching. The lineplots show the mean throughput.



(a) BlueStore backend

(b) POSIX backend

Figure 5. Throughput for writing without explicit syncing for BlueStore and POSIX. All operations are evaluated with and without batching. The lineplots show the mean throughput.

The general JULEA architecture is shown in Figure 3. When using JULEA, the application interacts with one or several JULEA clients that communicate with the appropriate backends. The backends are managed by server processes on the appropriate nodes. By offering a generic interface for every one of these concepts the actual implementations can be easily exchanged. So it is possible, just by adapting the configuration, to switch the database backend, e.g. from SQLite to MariaDB. Moreover, the backends can be deployed at different hardware levels, increasing the versatility of JULEA considerably.

Currently, the object store is a prototype built to store the data directly in POSIX which brings along the expected limitations discussed in the introduction. Objects are mapped to files similar to Ceph's FileStore.

BlueStore Library and Backend. To ease the future usage of BlueStore independently of Ceph, we developed a small library providing the general functionality. The following calls are supported at the moment: `init`, `mkfs`, `mount`, `create collection`, `umount`, `create`, `delete`, `write`, `read`, `status` The JULEA backend for BlueStore is in essence a thin layer wrapping the library functions. Thus, the object store client uses the BlueStore backend which in turn uses the BlueStore Library.

How To. To use BlueStore with JULEA, the user needs to be able to either install the Ceph dependencies themselves or convince the admin of their cluster to provide them. Then, Ceph needs to be compiled to obtain the necessary BlueStore libraries. We run BlueStore on a loop device as block storage to avoid having to repartition the existing storage devices. Even though the current state is still away from our final goal to extract BlueStore, it offers a considerable simplification compared to the only previous option of running a full-fledged Ceph cluster.

4 Evaluation

In the following, we first explain our setup and then we present the results of our evaluation of BlueStore. We compared JULEA using the BlueStore backend against JULEA using the POSIX-based object store. We used one compute node for now, to avoid conflicts between Ceph and the NFS of our cluster.

Hardware. The compute node is equipped with 4 × AMD Opteron 6344, 128 GB of main memory and a 1 TB WDC WD1003FBYZ-010FB0 HDD (with a maximum throughput of roughly 130 MB/s). For a baseline, we measured the HDD's

writing performance to be 109 MB/s using dd with a block size of 4k and the sync flag:

```
dd if=/dev/zero of=dummy bs=4k
    ↪ count=1000 oflag=sync
    ↪ status=progress
```

Listing 1. dd call for HDD performance baseline

Software. For the purpose of reproducibility, we list the version of the software we used below:

- OS + Kernel: Ubuntu 4.15.0-118-generic
- Ceph: master from 2021-02-13[5]
- JULEA BlueStore: from 2021-02-23[6]
- Compiler: GCC 9.3.0

Evaluation. As mentioned before, this evaluation is meant as a first step towards an understanding of BlueStore and its application areas. The results for both the BlueStore and the POSIX backend can be found in Figures 4 and 5. We performed measurements for various durations ranging from 1 second to 512 seconds, to rule out variability over time. As we found none, we present the results for a duration of 4 seconds per block size over a total of 10 runs for block sizes from 4 KiB to 4,096 KiB.

To avoid measuring only the cache, we set JULEA's storage semantics parameter to `storage=safety` for the runs in Figure 4, meaning that every operation is directly synced to the HDD. However, as this is very extreme behavior, we also evaluated batched JULEA operations. Depending on the block size, a sync is performed after every 10.000th operation for block sizes up to 256 kiB, respectively 1.000th operation for all larger block sizes.

This difference does not have an impact on the performance as can be seen in all plots. As the results for synced and unsynced reading are very similar for both backends, we only show the synced results for brevity's sake. The resemblance is no surprise, as the explicit syncing in JULEA does not considerably change the internal behavior for reading.

Discussion. Synced writes without batching behave similarly in both POSIX and BlueStore. However, POSIX achieves about double BlueStore's performance, with POSIX reaching a peak performance of 50 MB/s for 4,096 KiB blocks whereas BlueStore only achieves 20.5 MB/s. Batching improves the synced writes for BlueStore and POSIX but only up to a block size of 512 kiB with a peak of 24.9 MB/s and 94 MB/s, respectively. The large drop afterwards to 10.3 MB/s for BlueStore and 28 MB/s for POSIX is unexpected and does not correspond to the change in batch size mentioned earlier. The exact reasons still have to be investigated. The unsynced results have their peak performance at either 512 kiB or 1024 kiB for all operations for both backends. There are caching influences that we could not circumvent yet, as can be seen for the reading performance. The BlueStore results vary wildly, which again points to cache interference. Nevertheless, the results are still meaningful as real workloads will also encounter cache interference.

[5] https://github.com/ceph/ceph/commit/71c33b8466d3af08d285896f42c9f12075d44091

[6] https://github.com/Bella42/julea/commit/bec331dae4e2c7c7183e6e45047368b8c28aa971

5 Conclusion and Future Work

We showed that BlueStore can be used as a semi-standalone object store. The evaluation showed that while there are still a lot of untapped potentials, our simple BlueStore backend works well. In the future, we want to decouple it from Ceph further, so that ideally not all Ceph dependencies need to be installed. Also, we will make use of more in-depth Ceph functionality to optimize the behavior. Furthermore, we will evaluate BlueStore's suitability to run across several nodes with JULEA.

Acknowledgments

This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 417705296. More information about the CoSEMoS (Coupled Storage System for Efficient Management of Self-Describing Data Formats) project can be found at https://cosemos.de.

We also thank our student Johannes Coym for his work on the BlueFS library and the JULEA backend.

References

[1] Abutalib Aghayev, Sage Weil, Greg Ganger, and George Amvrosiadis. 2019. *Reconciling LSM-Trees with Modern Hard Drives using BlueFS.* Technical Report. Technical Report CMU-PDL–, CMU Parallel Data Laboratory. https://www.pdl.cmu.edu/PDL-FTP/FS/CMU-PDL-19-102.pdf

[2] Abutalib Aghayev, Sage A. Weil, Michael Kuchnik, Mark Nelson, Gregory R. Ganger, and George Amvrosiadis. 2019. File systems unfit as distributed storage backends: lessons from 10 years of Ceph evolution. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP 2019, Huntsville, ON, Canada, October 27-30, 2019*, Tim Brecht and Carey Williamson (Eds.). ACM, 353–369. https://doi.org/10.1145/3341301.3359656

[3] Abutalib Aghayev, Sage A. Weil, Michael Kuchnik, Mark Nelson, Gregory R. Ganger, and George Amvrosiadis. 2020. The Case for Custom Storage Backends in Distributed Storage Systems. *ACM Trans. Storage* 16, 2 (2020), 9:1–9:31. https://doi.org/10.1145/3386362

[4] Xiaowei Chu, Jeff LeFevre, Aldrin Montana, Dana Robinson, Quincey Koziol, Peter Alvaro, and Carlos Maltzahn. 2020. Mapping Datasets to Object Storage System. *CoRR* abs/2007.01789 (2020). arXiv:2007.01789 https://arxiv.org/abs/2007.01789

[5] Matthias Grawinkel, Lars Nagel, Markus Mäsker, Federico Padua, André Brinkmann, and Lennart Sorth. 2015. Analysis of the ECMWF Storage Landscape. In *Proceedings of the 13th USENIX Conference on File and Storage Technologies, FAST 2015, Santa Clara, CA, USA, February 16-19, 2015*, Jiri Schindler and Erez Zadok (Eds.). USENIX Association, 15–27. https://www.usenix.org/conference/fast15/technical-sessions/presentation/grawinkel

[6] Michael Kuhn. 2017. JULEA: A Flexible Storage Framework for HPC. In *High Performance Computing - ISC High Performance 2017 International*

Workshops, DRBSD, ExaComm, HCPM, HPC-IODC, IWOPH, IXPUG, P^3MA, VHPC, Visualization at Scale, WOPSSS, Frankfurt, Germany, June 18-22, 2017, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 10524), Julian M. Kunkel, Rio Yokota, Michela Taufer, and John Shalf (Eds.). Springer, 712–723. https://doi.org/10.1007/978-3-319-67630-2_51

[7] Michael Kuhn and Kira Duwe. In press. Coupling Storage Systems and Self-Describing Data Formats for Global Metadata Management. In *International Conference on Computational Science and Computational Intelligence (CSCI 2020)*. Conference Publishing Services (CPS).

[8] Dong-Yun Lee, Kisik Jeong, Sang-Hoon Han, Jin-Soo Kim, Joo-Young Hwang, and Sangyeun Cho. 2017. Understanding write behaviors of storage backends in ceph object store. In *Proceedings of the 2017 IEEE International Conference on Massive Storage Systems and Technology*, Vol. 10.

[9] Eunji Lee, Youil Han, Suli Yang, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. 2018. How to Teach an Old File System Dog New Object Store Tricks. In *10th USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage 2018, Boston, MA, USA, July 9-10, 2018*, Ashvin Goel and Nisha Talagala (Eds.). USENIX Association. https://www.usenix.org/conference/hotstorage18/presentation/lee

[10] Zhen Liang, Johann Lombardi, Mohamad Chaarawi, and Michael Hennecke. 2020. DAOS: A Scale-Out High Performance Storage Stack for Storage Class Memory. In *Supercomputing Frontiers - 6th Asian Conference, SCFA 2020, Singapore, February 24-27, 2020, Proceedings (Lecture Notes in Computer Science, Vol. 12082)*, Dhabaleswar K. Panda (Ed.). Springer, 40–54. https://doi.org/10.1007/978-3-030-48842-0_3

[11] Kunal Lillaney, Vasily Tarasov, David Pease, and Randal C. Burns. 2019. Agni: An Efficient Dual-access File System over Object Storage. In *Proceedings of the ACM Symposium on Cloud Computing, SoCC 2019, Santa Cruz, CA, USA, November 20-23, 2019*. ACM, 390–402. https://doi.org/10.1145/3357223.3362703

[12] Kunal Lillaney, Vasily Tarasov, David Pease, and Randal C. Burns. 2019. The Case for Dual-access File Systems over Object Storage. In *11th USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage 2019, Renton, WA, USA, July 8-9, 2019*, Daniel Peek and Gala Yadgar (Eds.). USENIX Association. https://www.usenix.org/conference/hotstorage19/presentation/lillaney

[13] Jingqing Mu, Jérome Soumagne, Suren Byna, Quincey Koziol, Houjun Tang, and Richard Warren. 2020. Interfacing HDF5 with a scalable object-centric storage system on hierarchical storage. *Concurr. Comput. Pract. Exp.* 32, 20 (2020). https://doi.org/10.1002/cpe.5715

[14] Jingqing Mu, Jérome Soumagne, Houjun Tang, Suren Byna, Quincey Koziol, and Richard Warren. 2018. A Transparent Server-Managed Object Storage System for HPC. In *IEEE International Conference on Cluster Computing, CLUSTER 2018, Belfast, UK, September 10-13, 2018*. IEEE Computer Society, 477–481. https://doi.org/10.1109/CLUSTER.2018.00063

[15] Simon D. Smart, Tiago Quintino, and Baudouin Raoult. 2019. A High-Performance Distributed Object-Store for Exascale Numerical Weather Prediction and Climate. In *Proceedings of the Platform for Advanced Scientific Computing Conference, PASC 2019, Zurich, Switzerland, June 12-14, 2019*. ACM, 16:1–16:11. https://doi.org/10.1145/3324989.3325726

[16] Houjun Tang, Suren Byna, Stephen Bailey, Zarija Lukic, Jialin Liu, Quincey Koziol, and Bin Dong. 2019. Tuning Object-Centric Data Management Systems for Large Scale Scientific Applications. In *26th IEEE International Conference on High Performance Computing, Data, and Analytics, HiPC 2019, Hyderabad, India, December 17-20, 2019*. IEEE, 103–112. https://doi.org/10.1109/HiPC.2019.00023

[17] Marc-André Vef, Rebecca Steiner, Reza Salkhordeh, Jörg Steinkamp, Florent Vennetier, Jean-François Smigielski, and André Brinkmann. 2020. DelveFS - An Event-Driven Semantic File System for Object Stores. In *IEEE International Conference on Cluster Computing, CLUSTER 2020, Kobe, Japan, September 14-17, 2020*. IEEE, 35–46. https://doi.org/10.1109/CLUSTER49012.2020.00014