Exploring Untrusted Distributed Storage for High Performance Computing

Austin Smith
Harvard University FAS Research Computing
austin_smith@harvard.edu

Justin Riley
Harvard University FAS Research Computing
justin_riley@harvard.edu

Muneeba Syed
Harvard University FAS Research Computing
muneeba_syed@harvard.edu

Milan Kupcevic
Harvard University FAS Research Computing
milan_kupcevic@harvard.edu

Paul Edmon
Harvard University FAS Research Computing
pedmon@cfa.harvard.edu

Scott Yockel
Harvard University FAS Research Computing
scott_yockel@harvard.edu

ABSTRACT

High performance computing systems are typically built with high-throughput and infrastructural uniformity in mind, but generally do not easily accommodate diverse data security requirements on a single cluster. Rather than fracturing that infrastructure by building many network isolated storage "islands" to secure each dataset covered by an individual data use agreement, we explore using the Ceph distributed storage system with client-side encryption to provision secure storage from a single, untrusted data lake.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**;

KEYWORDS

ceph, high performance computing, encryption, distributed storage

ACM Reference Format:
Austin Smith, Justin Riley, Muneeba Syed, Milan Kupcevic, Paul Edmon, and Scott Yockel. 2019. Exploring Untrusted Distributed Storage for High Performance Computing. In *Practice and Experience in Advanced Research Computing (PEARC'19), July 28-August 1, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3332186.3332224

1 INTRODUCTION

As the volume and velocity of data generated in the course of everyday life continues to accelerate, researchers across various disciplines are increasingly turning to computational methods in order to gain quantitative insights from new data sets. However, this abundance of data poses a challenge for research computing centers. Researchers frequently want to work with sensitive data sets containing personally identifiable or other private information, whose use is governed by specific security requirements. Since these requirements are often mutually exclusive from one data use agreement to the next, it can be difficult to safeguard research subject data while also providing a uniform and performant computing infrastructure where researchers can conduct analysis at scale.

We present a novel method for deploying Ceph [19] object gateways [6] on dedicated compute nodes, in order to perform encryption client-side, treating the Ceph storage cluster as an untrusted shared resource. Our approach limits the exposure of plaintext data to dedicated, single tenant compute nodes. By reducing the footprint of the trusted computing base and relying on encryption, rather than network isolation, to partition storage security boundaries, users may benefit from a performant, distributed, and elastic shared storage system for their secure computing, instead of making do with individual RAID-based storage, provisioned on a per-project basis, which lacks the horizontal scalability and performance of modern distributed storage systems.

2 BACKGROUND

Traditionally, high performance computing clusters are treated as uniform security environments. Isolated from public networks, it is assumed that users who can connect and authenticate are authorized to access the system. Infrastructure operators are trusted by default. Individual users are isolated from one another by ordinary POSIX permissions and unprivileged user accounts. When robust security is required, the standard solution is to build several physically isolated clusters and assign each one a different security level. This approach works well in classified environments, where the levels of security are clearly defined and where an entire "secure" cluster is used by a single tenant at a time.

However, for multi-tenant high performance computing platforms, such as those found at research universities and in the financial technology sector, where users run jobs concurrently and security requirements are both less uniform and typically fall short of requiring physical isolation, operating a dedicated "secure" cluster that can only be used by one tenant at a time is impractical. Instead, a typical approach is to create various isolated environments via network partitions. However, partitioning computing resources into isolated environments necessarily reduces the throughput available to any one user, since a hypothetical user in a "secure" environment cannot use idle resources in the general "non-secure" environment, and *vice versa*. Partitioning storage is similarly cumbersome. Users

PEARC '19, July 28-August 1, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-7227-5/19/07.
https://doi.org/10.1145/3332186.3332224

are stuck with static allocations of network isolated "secure" storage systems that do not scale elastically, while administrators must contend with managing a fractured infrastructure of such storage "islands," which lack the uniformity, elasticity, and performance of modern distributed storage systems.

Distributed storage systems have been integral to high performance computing since the decline of monolithic supercomputer systems and the emergence of distributed computation on commodity hardware with high-speed interconnects. Compared to more familiar open source distributed storage systems, such as Lustre, Ceph is a relative newcomer. While Lustre was designed as a distributed file system with high performance computing applications in mind, Ceph was conceived as a general-purpose distributed object store, capable of presenting block, file, and object storage abstractions from the same cluster, making it suitable as a storage backend for cloud computing infrastructure, a distributed file system for high performance computing, or as an object store similar to Amazon S3. Despite being a relative newcomer, Ceph has gained increasing interest and adoption among research computing centers for its versatility, robustness, and horizontal scalability. CERN makes extensive use of Ceph for research computing, operating a production cluster exceeding sixty-four petabytes [2]. Harvard FAS Research Computing has built production OpenNebula cloud infrastructure on top of Ceph [16] and is currently deploying NESE [14], a large multi-institutional Ceph cluster at the MGHPCC datacenter, in partnership with Boston University, MIT, Northeastern University, and the University of Massachusetts.

Since the POSIX file system is the *lingua franca* of HPC storage, and since Lustre, Ceph, and other distributed storage systems all present a POSIX interface, the most straightforward means to enable client-side encryption would appear to be simply layering existing disk encryption solutions on top of whichever distributed file system is already employed on-site. Tools such as dm-crypt and eCryptfs are widely used and are already included in the Linux kernel.

Unfortunately, problems with this approach arise immediately. Since dm-crypt is designed to encrypt block storage, which cannot be mounted to more than one machine at a time, it is unsuitable for encrypting a distributed file system that may be mounted to many machines at once, in an HPC environment. As a file-based disk encryption system, eCryptfs initially appears more promising, but unfortunately suffers from long-standing bugs when used on top of a network file system [7]. FUSE-based encrypted file systems, such as EncFS and gocryptfs [10] do work well over network file systems, but both allow users to inadvertently write unencrypted data directly to the network file system, effectively bypassing the protections offered by encrypting their storage in the first place. EncFS also suffers from several publicly known and unaddressed security vulnerabilities [9]. There does not appear to be a suitable, open source encrypted file system that offers encryption by default and which performs well on shared storage over network file systems. Fortunately, due to the versatility of Ceph, it is possible to encrypt *object* storage client-side, before the data is transmitted to the storage cluster.

3 DESIGN

At a high level, a Ceph cluster is composed of monitor daemons (monitors) and object storage daemons (OSDs). Monitors maintain a master copy of the cluster map. They are typically deployed in odd numbers, with one monitor daemon each on a dedicated physical machine, and maintain state consensus for high availability via the PAXOS algorithm [12]. A typical cluster has three monitors. Object storage daemons are deployed for every physical drive in the cluster and act as intermediaries between the physical disk resource they govern and the distributed object store abstraction they present over the network.

Ceph exposes four interfaces for clients to access storage resources:

- CephFS, a POSIX compliant distributed file system,
- RBD, a distributed block device,
- librados, a library allowing applications to access the object store directly,
- and the Ceph Object Gateway (or RADOS [21] Gateway), a bucket-based REST API gateway compatible with Amazon S3.

The Ceph Object Gateway is deployed as an HTTP server interface to Ceph, with multiple clients making API calls to the gateway or gateways, which themselves act as librados clients to the cluster, writing directly to the OSDs. Much like Amazon S3, the Ceph Object Gateway supports server-side encryption of objects, with customer provided keys. Typically these requests are made over TLS, with clients sending their encryption key to the gateway along with each request to read or write an object. Since encryption typically occurs server-side, clients have no control over whether encryption takes place at all and necessarily expose their keys to a centralized service. Therefore, with server-side encryption, the user's data is encrypted at the gateway, before it is written to the Ceph cluster as ciphertext.

Server-side encryption at the Ceph Object Gateway has two principal drawbacks. First, the client must trust the server to perform encryption and handle their encryption keys for them. This is obviously suboptimal from a security point of view. Second, the object gateway poses a network bottleneck for clients writing to the Ceph cluster. One of the principal advantages of Ceph is that clients can write directly to OSDs by calculating where to store and retrieve data deterministically, via the CRUSH algorithm [20] and the cluster map, without being burdened by the network bottleneck of first connecting to a metadata server. By interposing itself between clients and OSDs, the Ceph Object Gateway re-introduces a network bottleneck, inhibiting performance.

However, both of these concerns can be resolved by moving the Ceph Object Gateways to the clients themselves. Since the object gateway is just an HTTP server exposing the S3 API, client applications can interact with that API over localhost just as well as over a network connection. By moving the gateways to the clients, "server-side" encryption becomes a client-side operation, so that clients can control their encryption keys directly. Distributing the object gateways on the clients also enables clients to interact directly with OSDs, eliminating the bottleneck posed by many clients accessing Ceph via an intermediary HTTP gateway server.

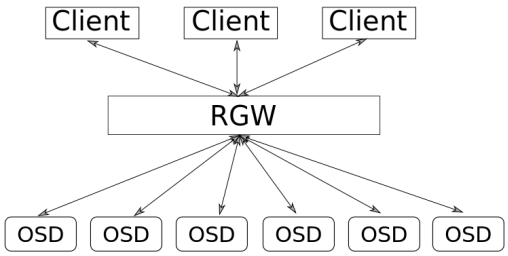

Figure 1: Clients connecting through a Ceph Object Gateway

From an application perspective, a useability problem remains: to use the S3 API, applications must be modified to employ object storage. In an enterprise environment, modifying a single application to integrate with S3 might be an acceptable burden, but for research computing centers, where users expect access to a library of hundreds, if not thousands of scientific applications, modifying applications to use the S3 API is non-viable. Fortunately, tools now exist to present S3 buckets as FUSE mounts, which incorporate a large subset of the POSIX standard that most HPC applications expect. As an integrated system, this configuration enables client machines to write ciphertext objects directly to an untrusted distributed storage system, while presenting a near-POSIX local filesystem to the application, which has no knowledge that data written to this filesystem is encrypted before leaving the client node. This ensures that the trusted computing base is entirely client-side, where data is processed on dedicated, single-tenant compute nodes, without burdening the user with key management or enabling them to accidentally violate the security boundary by writing plaintext data directly to the untrusted, shared Ceph cluster. Thus, the user's plaintext data is only exposed to their client-side compute nodes, when their data is processed by the application.

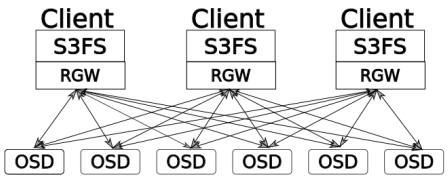

Figure 2: Ceph Object Gateways on Clients connecting directly to OSDs

4 IMPLEMENTATION

To test our client-side encryption concept under simulated production loads, we made use of the NESE Ceph cluster at the MGHPCC data center, as well as 1,696 cores of compute, distributed across 106 client nodes. The NESE cluster consists of three monitors and 702 object storage daemons (one daemon per disk), spread across 57 OSD nodes. For these tests, our storage pool was configured with (4, 2) erasure coding. [15] Our test compute consisted of dual socket, 16 core machines, running Intel Xeon E5520 processors with a base clock speed of 2.27 GHz.

We deployed a Ceph Object Gateway in a Linux container on each compute client and chose S3FS [17] to present a POSIX interface to the local client-side application. S3FS ordinarily translates a near-POSIX local filesystem interface to S3 API calls over the network. In a typical configuration with Ceph, S3FS would be installed on the client and would make calls to one of a few Ceph Object Gateways in order to interact with the storage cluster. This is convenient when client-side encryption is not a requirement and when clients are not located directly on the Ceph cluster network, since clients can rely on the Ceph Object Gateway to perform encryption server-side and use the S3 API to interact with the storage cluster over wide area networks. This is the same workflow that applications use to interact with Amazon S3; clients access the storage via gateways that expose the S3 API. However, in HPC clusters where compute and storage typically enjoy physical proximity and network adjacency, it is preferable to write directly to the OSDs, rather than funneling requests through a few gateways. In our configuration, both the Ceph Object Gateway and S3FS are deployed in containers on the client nodes and communicate with one another over localhost, both in order to perform encryption client-side and to eliminate the network bottleneck posed by a small number of gateways interposing themselves between the hundreds of clients and OSDs that may need to communicate in parallel in a distributed computing system.

With both a Ceph Object Gateway and 3SFS deployed on all clients in Linux containers, each client has access to the Ceph cluster, via a shared S3 bucket, which is presented as a filesystem locally by S3FS. Although the S3FS mount on each client could be exposed either inside the S3FS container or outside of the container, on the host file system, we chose the latter route. By mounting S3FS to a bind mount shared with the host, inside a Linux container volume mapped to the underlying host's filesystem, the "locally" mounted network storage appears as an ordinary directory to applications and users on the host operating system, outside of the S3FS container.

We see several benefits to exposing the storage as an ordinary bind mount on the compute node's host operating system. First, it completely abstracts away the file-to-object translation performed by S3FS, as well as the client-side encryption performed by the Ceph Object Gateway running on a separate container on the same machine. Since users and their applications typically do not have privileged root access in HPC computing environments, mapping the S3FS mount point to a directory on the host system prevents users from accidentally writing unencrypted data to the untrusted Ceph cluster, since the encryption keys and S3 credentials needed to access the storage reside inside the S3FS container. Second, exposing the mount to the host ensures compatibility with existing HPC tooling. Unmodified scientific applications may be loaded through a traditional module system and managed via the existing scheduler. Rather than re-building these applications as containers, this approach presents secure storage in a format the application already expects and with which users are already familiar: an ordinary directory on the host system. However, unlike other network file

systems, everything they write is encrypted on the client and the storage cluster is never trusted with the content of their data.

Table 1: CephFS Performance

Operation	Max IOPS	Min IOPS	Mean IOPS
Directory creation:	3363.190	3363.168	3363.180
Directory stat:	8817800.917	5190063.153	6409776.669
Directory removal:	5671.911	5671.295	5671.632
File creation:	6370.684	6370.576	6370.623
File stat:	43128212.571	4469472.802	17613571.855
File read:	36037558.707	4201603.747	25084764.971
File removal:	5777.656	5777.511	5777.573
Tree creation:	703.679	569.761	638.321
Tree removal:	14.873	8.222	11.864

5 PERFORMANCE

In order to establish a baseline for performance, we first compared our client-side encrypted S3FS implementation to CephFS. CephFS is the standard POSIX mode for Ceph, so it serves as a good initial point of comparison. We expected CephFS to outperform our S3FS implementation, both because of the overhead imposed by encryption, as well as the additional overhead of translating POSIX to S3 and S3 to librados, on every client. Nevertheless, this test was a good basis for us to assess whether our configuration is a usable filesystem under saturating loads and to compare it to a known system that is already used by some, in production.

For benchmarking, we used MDTest, [13] a message passing interface-based application for evaluating file system metadata performance. MDTest measures performance by creating, stat-ing, and deleting a tree of directories and files across a cluster of machines. MDTest relies on a message passing interface (MPI) to orchestrate parallel tasks. MDTest and OpenMPI [8] enabled us to distribute the file operations across 106 client machines and use all 1,696 cores to simulate parallel file system load. We were unsurprised to find that CephFS (*see: Table 1*) is significantly more performant than our S3FS implementation (*see: Table 2*), given the additional overhead our system entails, but we were encouraged to see that the system remained usable despite distributed file operations performed by 1,696 threads in parallel.

Table 2: S3FS Performance

Operation	Max IOPS	Min IOPS	Mean IOPS
Directory creation:	645.778	645.776	645.777
Directory stat:	61025.342	60987.303	61002.794
Directory removal:	721.686	721.685	721.685
File creation:	565.436	565.433	565.434
File stat:	18744.482	18742.516	18743.708
File read:	116.670	116.670	116.670
File removal:	3870.483	3870.462	3870.472
Tree creation:	6.174	5.259	5.769
Tree removal:	8.946	8.512	8.714

6 DYNAMIC ALLOCATION OF SECURE STORAGE & COMPUTE

In our discussion of secure storage thus far, we have largely assumed a static allocation of single-tenant compute nodes accessing an elastic, untrusted storage cluster. This isolation of single-tenant machines is important in a multi-tenant environment, since even though the untrusted storage cluster doesn't have access to users' plaintext, their data remains vulnerable at runtime. Nevertheless, static allocations of single-tenant compute remain undesirable, for two reasons. First, idle cycles on a given user's "secure" compute cannot be used by other researchers on the same cluster. This undermines the fairshare model employed by many research computing centers, where users get exclusive priority to run jobs on compute they own, but other researchers can make use of their compute as well, provided it is idle. Dedicated single-tenancy, while a security improvement, necessarily undermines this principle. Second, static allocations of compute restrict a researcher's computing power to what they can afford to buy outright. With the fairshare model, users can "borrow" vast computing capacity for a short time, far beyond what they may have purchased. This isn't possible if researchers are required to run all of their jobs on a few machines, for security purposes.

Since we are naturally interested in enabling users to run secure jobs at scale, we built a proof of concept using Singularity containers, [11] in order to take advantage of a new feature of the Slurm Workload Manager, called Multi-Category Security (MCS). [18] In the initial test environment that we used for benchmarking, we employed Docker to deploy Ceph Object Gateways and S3FS. This worked well for a block of isolated compute with no other users present. However, in a shared computing environment, exposing sensitive data across the cluster is obviously unacceptable, since any user who is able to escalate their privileges to root would be able to read and exfiltrate data from the encrypted S3FS mount points, since these mounts expose the plaintext to the client machines, where the data is only protected by POSIX permissions.

In order to prevent this sort of security breach, it is clearly important to be able to stand up and tear down the containers that expose the secure storage on demand, wherever the user runs their secure jobs. Ideally the user could do this themselves. This naturally leads to the notion that the Ceph Object Gateway and S3FS containers should be scheduled as ordinary jobs on the cluster, by the user. The first question that arises is how to ensure single-tenancy on any node where a secure mount is presented. If users may schedule their own secure jobs, then they must be the only user on a given compute node when their storage is mounted, to prevent potentially exposing their plaintext to malicious co-tenants. Slurm's MCS plugin offers exactly this functionality. With MCS, Slurm can reserve nodes for the exclusive use of a given user or group of users, in order to ensure single-tenancy at runtime, so that users' secure storage is only mounted to machines over which they have exclusive use. This ephemeral provisioning of secure storage and compute would enable researchers to perform large-scale computation using untrusted storage on single-tenant compute, without having to exclusively purchase dedicated hardware. However, in order to schedule secure storage mounts as jobs, an HPC-native container runtime is needed.

Singularity is a container runtime specifically developed to run in HPC environments. Unlike Docker, Singularity containers do not utilize a daemon process to manage running containers. This means that batch schedulers such as Slurm can manage Singularity containers as ordinary jobs, which run with the same user identifier (UID) and group identifier (GID) as the user who initiated the job. Additionally, Singularity makes it fairly straightforward to limit a user's ability to escalate privileges within a container. For our purposes, this is useful for keeping encryption and access keys away from the user, so that they cannot accidentally violate their data use agreement and write plaintext directly to Ceph, or elect to do so because they find that the performance is better. However, it would also be feasible to give a user access to Ceph but not administer their encryption keys at all, ensuring that only they retain access to their data, enabling them to leverage an untrusted elastic storage service without the burden of directly administering the storage themselves.

7 RELATED WORK

Neither client-side encrypted file systems nor the notion of deploying Ceph Object Gateways on clients are completely new ideas. Matt Blaze introduced CFS in 1993, [1] which used a client-side encryption engine alongside NFS mounts to clients' localhost interface in order to present a POSIX network file system where the remote storage media remained untrusted, during his time at Bell Labs. More recently, the Rutherford Appleton Laboratory in the UK has built "Echo" [5] a Ceph cluster exceeding 30 petabytes that primarily processes data for the Large Hadron Collider. Their initial architecture consisted of several external Ceph Object Gateways, which applications running on compute would connect to, in order to access storage. However, in a presentation at Cephalocon 2018,[3] they reported that this configuration quickly proved infeasible. Instead, they deployed the gateways in containers on all of compute. The applications running on these worker nodes then accessed storage via localhost calls to the gateway running on the same machine. RAL reported that this configuration was much more performant and expressed confidence in their distributed gateway model. However, neither of these systems combines modern distributed storage technology with client-side encryption to offer untrusted storage as a service for scientific computing.

8 CONCLUSION

Accommodating security requirements for sensitive data sets by fracturing cluster infrastructure into single-tenant, network partitioned "islands" of storage and compute is a loss for both HPC administrators and users. Such a Balkanized infrastructure is more difficult for administrators to maintain than a more uniform environment. Users, too, would be better served by elastic and horizontally scalable compute and storage resources that can be used to process sensitive data, instead of being limited to statically allocated resources in their "secure" environment. Such partitions also erode the overall throughput of a computing resource, since users in isolated environments cannot take advantage of idle resources elsewhere on the same cluster, as they ordinarily would be able to do on a uniform system.

We may say that for security requirements short of physical isolation in a multi-tenant environment, an ideal high performance computing cluster would isolate users' processes and data from other users, as well as from the infrastructure operators, without sacrificing overall throughput, all the while enabling users to run ordinary jobs on non-sensitive data without these security features, on the same infrastructure. Such a system could consist of trusted execution environments for running isolated processes, which could mount and decrypt untrusted distributed storage. Recent generations of Intel CPUs include hardware and software security features for deploying such trusted execution environments, called SGX "enclaves," [4] on untrusted hardware. Since Intel SGX incorporates public key infrastructure enabling remote attestation, and since these enclaves aim to protect processes from an untrusted kernel and system BIOS, this would meet the requirements for our ideal untrusted multi-tenant HPC cluster. However, before such a system can be built, the more basic problem of client-side storage encryption must be addressed.

Solving these issues while ensuring that sensitive data is safely isolated is difficult. While secure computing "enclaves" such as Intel SGX, combined with in-enclave encryption of distributed storage, could dramatically reduce the scope of the trusted computing base for HPC and enable comparatively elastic and reasonably secure multi-tenant computing, the state of client-side encryption for distributed storage systems must advance first. Our method for deploying Ceph Object Gateways on clients makes use of existing server-side encryption implementations to instead offer client-side encryption, enabling users to treat a shared, distributed storage system as an untrusted, elastic resource.

REFERENCES

[1] Matt Blaze. 1993. A Cryptographic File System for UNIX. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS '93)*. ACM, New York, NY, USA, 9–16. https://doi.org/10.1145/168588.168590

[2] Ceph Blog. [n. d.]. Ceph Blog: New in Luminous: Improved Scalability. https://ceph.com/community/new-luminous-scalability/

[3] Thomas William Byrne. [n. d.]. Erasure Code at Scale. https://www.youtube.com/watch?v=cX8LlFZDfqU

[4] Victor Costan and Srinivas Devadas. 2016. Intel SGX Explained. *IACR Cryptology ePrint Archive* 2016 (2016), 86.

[5] Alastair Dewhurst. [n. d.]. Deployment of a large erasure coded object store for data intensive science. http://193.62.125.70/CIUK2017/AlastairDewhurst_STFC.pdf

[6] Ceph Documentation. [n. d.]. Ceph Blog Object Gateway. http://docs.ceph.com/docs/mimic/radosgw/

[7] eCryptfs Bug Tracker. [n. d.]. ecryptfs does not work properly over nfs, cifs, samba, WebDAV, or aufs. https://bugs.launchpad.net/ecryptfs/+bug/277578

[8] Richard Graham, Timothy S. Woodall, and Jeffrey Squyres. 2005. Open MPI: A flexible high performance MPI. 228–239. https://doi.org/10.1007/11752578_29

[9] Taylor. Hornby. 2014. *EncFS Security Audit.* Technical Report.

[10] Taylor. Hornby. 2017. *Security Audit of gocryptfs v1.2.* Technical Report.

[11] Sochat V Kurtzer GM and Bauer MW. 2017. Singularity: Scientific containers for mobility of compute. *Computer* (2017). https://doi.org/doi:10.1371/journal.pone.0177459

[12] Leslie Lamport. 1998. The Part-time Parliament. *ACM Trans. Comput. Syst.* 16, 2 (May 1998), 133–169. https://doi.org/10.1145/279227.279229

[13] mdtest. [n. d.]. mdtest. https://github.com/hpc/ior

[14] NESE. [n. d.]. NESE Overview. http://nese.mghpcc.org/about/

[15] James. Plank. 2013. *Erasure Codes for Storage Systems.* Technical Report.

[16] J. Riley, J. Noss, W. Dillingham, J. Cuff, and I. M. Llorente. 2017. A High-Availability Cloud for Research Computing. *Computer* 50, 6 (2017), 92–95. https://doi.org/10.1109/MC.2017.182

[17] s3fs. [n. d.]. s3fs. https://github.com/s3fs-fuse/s3fs-fuse

[18] Slurm. [n. d.]. Slurm Multi-Category Security. https://slurm.schedmd.com/mcs.html

[19] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn. 2006. Ceph: A Scalable, High-Performance Distributed File System. In *OSDI*.

[20] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, and Carlos Maltzahn. 2006. CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (SC '06)*. ACM, New York, NY, USA, Article 122. https://doi.org/10.1145/1188455.1188582

[21] Sage A. Weil, Andrew W. Leung, Scott A. Brandt, and Carlos Maltzahn. 2007. RADOS: A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters. In *Proceedings of the 2Nd International Workshop on Petascale Data Storage: Held in Conjunction with Supercomputing '07 (PDSW '07)*. ACM, New York, NY, USA, 35–44. https://doi.org/10.1145/1374596.1374606