Federated Access from DOE Labs to Distributed Storage in the EIC Era of Computing

M D Poat[1], J Lauret[1] , T Rao[1]
[1] Brookhaven National Laboratory, P.O Box 5000, Upton, New York 11973-5000, USA

mpoat@bnl.gov , jlauret@bnl.gov, raot@bnl.gov

Abstract. The Electron Ion Collider (EIC) collaboration and future experiment is a unique scientific ecosystem within Nuclear Physics as the experiment starts right off as a cross-collaboration between Brookhaven National Lab (BNL) & Jefferson Lab (JLab). As a result, this muti-lab computing model tries at best to provide services accessible from anywhere by anyone who is part of the collaboration. While the computing model for the EIC is not finalized, it is anticipated that the computational and storage resources will be made accessible to a wide range of collaborators across the world. The use of federated ID seems to be a critical element to the strategy of providing such services, allowing seamless access to each lab site computing resources. However, providing Federated access to a Federated storage is not a trivial matter and has its share of technical challenges.

In this contribution, we focus on the steps we took towards the deployment of a distributed object storage system that integrates with Amazon S3 and Federated ID. We will first cover for and explain the first stage storage solutions provided to the EIC during the detector design phase. Our initial test deployment consisted of Lustre storage using MinIO, hence providing an S3 interface. High Availability load balancers were added later to provide the initial scalability it lacked. Performance of that system will be shown. While this embryonic solution worked well, it had many limitations. Looking ahead, the Ceph object storage is considered a top-of-the-line solution in the storage community - since the Ceph Object Gateway is compatible with the Amazon S3 API out of the box, our next phase will use a native S3 storage. Our Ceph deployment will consist of erasure coded storage nodes to maximize storage potential along with multiple Ceph Object Gateways for redundant access. We will compare performance of our next stage implementations. Finally, we will present how to leverage OpenID Connect with the Ceph Object Gateway's to enable Federated ID access.

We hope this contribution will serve the community needs as we move forward with cross-lab collaborations and the need for Federated ID access to distributed compute facilities.

1. Introduction

The Electron-Ion Collider (EIC) will stand as the next generation of Nuclear Physics collider at a flagship facility for nuclear physics research. The EIC is set to be anchored at the Brookhaven National Laboratory (BNL) and this groundbreaking initiative represents a synergistic alliance between BNL and the Thomas Jefferson National Accelerator Facility (JLab). As of this writing, the computing blueprint for the EIC remains in the conceptual stage, though there's a collective aspiration to render storage resources available to an extensive set of international collaborators. Such an endeavour underscores the necessity for a Federated storage architecture, harmonized with Federated ID access, ideally echoing S3 compatibility standards or similar. Supported by a "Program Development" funding, we spearheaded an S3 demonstrator project that bifurcated into two distinct implementations. Our initial setup revolved around an S3 demonstrator, instrumented by a MinIO Gateway [1]. This delivered an Object Storage like interface backed by Lustre POSIX storage. Subsequently, our efforts culminated in the deployment of an authentic Object Storage solution using Ceph, complemented by the Ceph Object Gateway [2].

2. MinIO Gateway

MinIO is a distributed object storage solution that provides an S3-compatible API and supports S3. While MinIO provides multiple methods to deploy a MinIO object storage, they also provide a service called the MinIO Gateway (Minio GW) which provides an S3 interface to a backend POSIX storage such as GPFS, NFS, Lustre, etc. At BNL we have an existing 3 PB Lustre Storage that we used for our MinIO GW setup. Our Lustre storage is made up of 3 hosts each consisting of 48 CPU Core, 392 GB RAM, 4x25 GbE links, and 100 14 TB drives. The MinIO version "RELEASE.2022-08-11" provides the MinIO GW service to create a quick and easy S3 interface to our Lustre storage. We first deployed the MinIO GW service on a single node, and while this lacked redundancy it did support work for the EIC community, once the service was requested to be permanent, we implemented a more robust setup.

The more robust setup followed using 4 hosts, each host consisting of 28 CPU Cores, 132GB Ram, and 4x25GbE links each. Two of the hosts, for simplicity's sake we will call them `eicdtn01` and `eicdtn02`, each run the services Keepalived [3] and HAProxy [4] and the other two hosts `eicdtn03` and `eicdtn04` run the MinIO GW service and mount Lustre. Keepalived and HAProxy are used in combination for failover and balancing. Keepalived implements the Virtual Router Redundancy Protocol (VRRP) [12] on a Linux System and manages the Linux Virtual Server Configuration. For example, if `eicdtn01` or `eicdtn02` crash, the remaining hosts' Keepalived service will fail over the bad hosts virtual network configuration and assume it. Additionally, The HAProxy service ran on both hosts deploy a round robin that forwards IO from incoming clients to the two MinIO GW and Lustre mount hosts (`eicdtn03` and `eicdtn04`).

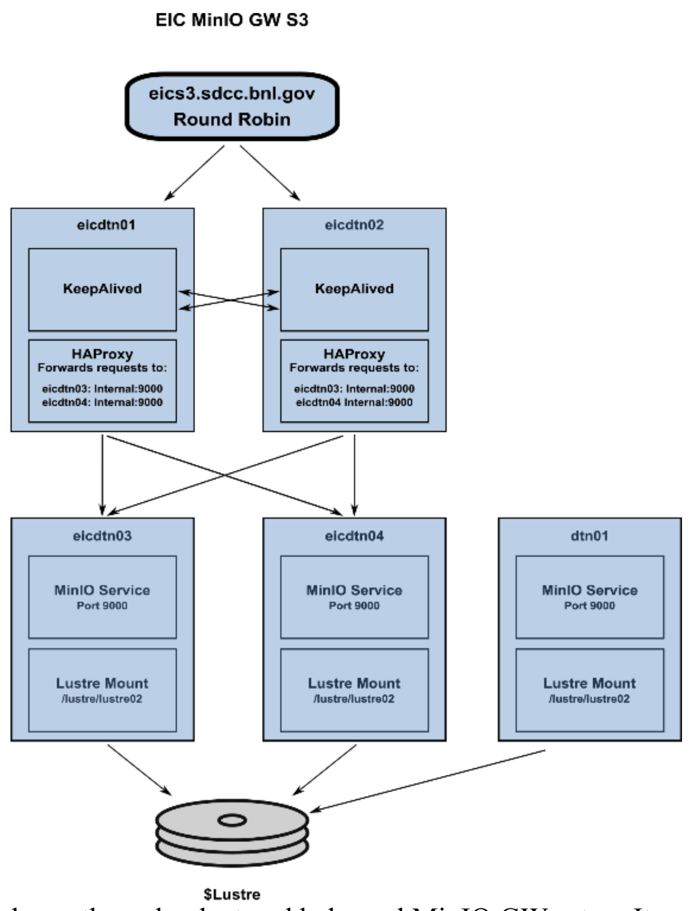


Figure 1. The diagram shows the redundant and balanced MinIO GW setup. It consists of 4 hosts, two running Keepalived and HAProxy, while the other two run the MinIO GW service and mount the Lustre Storage.

As it can be seen in Figure 1, our MinIO GW service provides more stability as it can sustain the loss of one Keepalived/HAProxy host and one MinIO GW/Lustre mount host. Though the MinIO GW configuration functions effectively within a singular site, it falls short in several areas, including zoning, Federated ID, and the capacity to expand across multiple data centers or to leverage multi-site replication. Moreover, MinIO disclosed in 2020 that the MinIO GW service was on the path to obsolescence. By 2022, their efforts were restricted to remedying bugs, with plans to completely phase out the service by 2023.

2. Ceph S3

Ceph is a reliable and scalable storage system based on RADOS (Reliable Autonomic Distributed Object Store) [6]. Ceph provides high availability, data protection, and supports features such as erasure coding and replication. The Ceph Object Gateway is the interface built on top of librados which provides the RESTful gateway between the Ceph storage cluster and the Amazon S3 API. Ceph was a first choice as we still support active Ceph clusters and have experience with deploying in the past. [7]

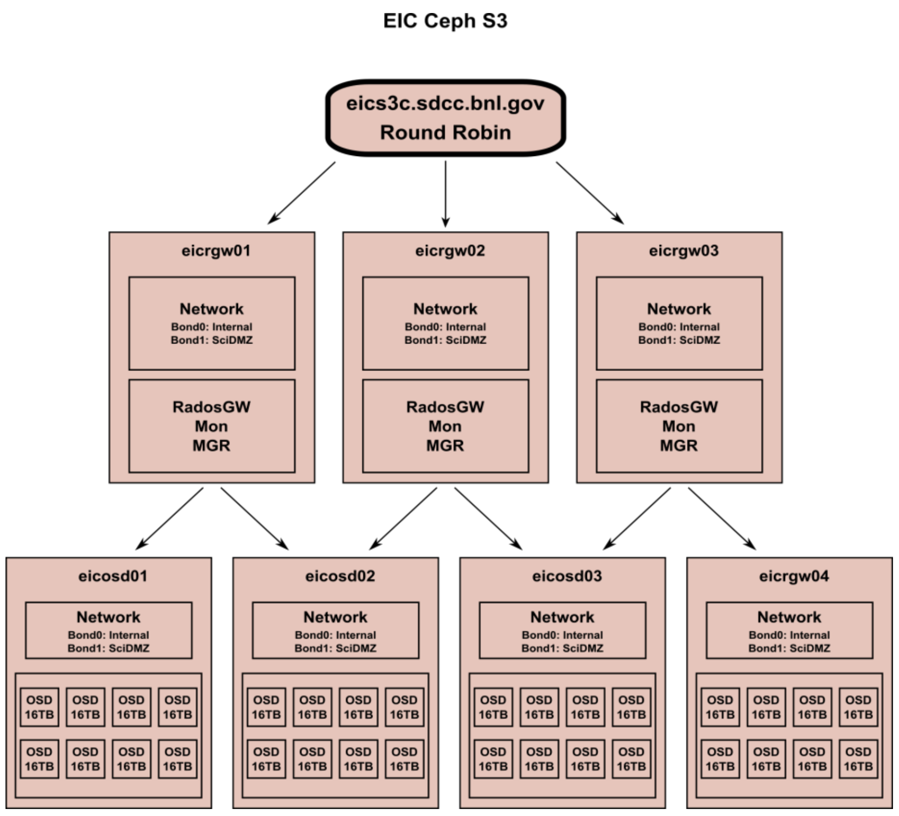


Figure 2. A diagram showing our EIC Ceph S3 cluster configuration consisting of 3 Rados Gateway hosts and 4 Ceph OSD hosts.

As shown in Figure 2, our initial S3 Ceph deployment consists of 3 hosts dedicated as the Rados Gateway servers along with the Ceph Monitor & Manager services. They are provisioned with 28 Cores, 256 GB RAM and 4x25 GbE connections. Additionally, we have 4 new OSD storage hosts, each host is provisioned with 48 Cores, 96 GB RAM, 4x25 GbE and 8x16 TB drives. This configuration provides us with around 450 TB of RAW storage. With Erasure Coding [8] 4+2 pools we end up with ~300 TB of usable redundant storage. This deployment is easily scalable, we can add disks to the current hosts or scale horizontally by adding more storage nodes.

3. Multi-Site Ceph

Ceph offers the ability to create a multi-site storage cluster that scales across multiple Ceph clusters in separate geographic locations. A multi-site Ceph cluster can be setup as multi-realm, multi-zone group or multi-zone. What's important to note is that you can replicate, mirror, or backup your data from one site to another.

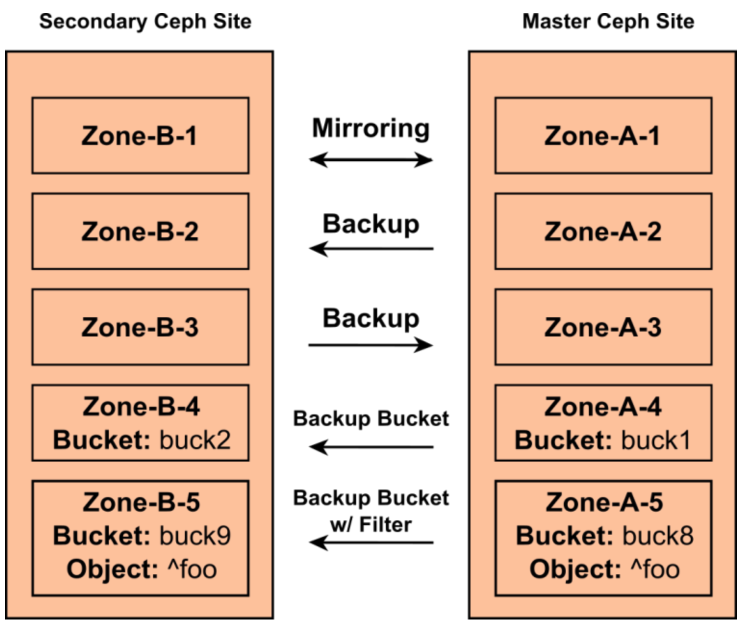


Figure 3. A diagram showing some of the data replication mechanisms that are available within a multi-site Ceph cluster.

As shown in Figure 3, a multi-site Ceph cluster can mirror two zones across sites. A user has the flexibility to write objects to any given zone. However, the metadata is exclusively written to the master zone. This approach facilitates swift local write operations and, crucially, circumvents potential metadata race conditions, given that the metadata consistently finds its place in the master zone. Users can also do bi-directional backups of data to and from either zone where the data is read-only on the secondary zone. Additionally, there is further granularity that enables users to do syncing at the Bucket level or even the bucket level with a filter (i.e sync objects that match a regular expression to/from Zone/Bucket). Overall, there are many mechanisms for syncing data across globally distributed Ceph cluster sites to meet many use cases.

4. Federated ID Access with OpenID Connect

A standard, unmodified setup of Ceph Object Storage is compatible with the Amazon S3 protocol. S3's authentication mechanism employs an ACCESS_KEY and a SECRET_KEY, which can be likened to a conventional username and password pairing. This is however not a secure method for distributed access as credentials could be shared or stolen. Ceph supports STS (Secure Token Service) a web service that enables you to request a temporary set of credentials to authenticate federated users [9]. Within STS there are many REST APIs for IAM (Identity Access Management), though within Ceph only `AssumeRole` and `AssumeRoleWithWebIdentity` STS APIs have been implemented. The `AssumeRole` API simply returns a set of temporary credentials that can be used for cross-account access. However, the `AssumeRoleWithWebIdentiy` API returns a set of temporary credentials for users that have been authenticated by an OpenID Connect /OAuth2.0 Identity Provider. Therefore, instead of using an assigned ACCESS_KEY & SECRET_KEY to authenticate in S3, you authenticate to an IDP (Identity Provider) which then returns a temporary Access token, an ACCESS_KEY,& SECRET_KEY. The token provides the authorization of the user, and these temporary credentials expire within a set time (hours). The benefit is this does not require owning any permanent credentials in S3 and the access is managed via the IDP. The access control becomes manageable and enables users from different institutions to authenticate. Currently only the Keycloak IDP has been tested and integrated within Ceph Rados Gateway.

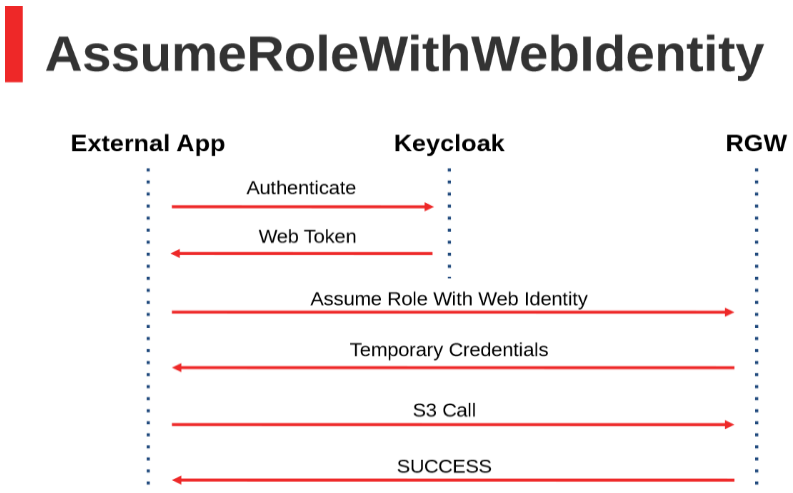


Figure 4. A diagram showing the steps to use `AssumeRoleWithWebIdentity` with STS

The steps to authenticate using STS and Keycloak are not obvious or well documented. After much testing we can provide details and steps on how to proceed. For clients, users can use the AWS CLI tool or the Python boto3 SDK as they both support `AssumeRoleWithWebIdentity`. As shown in Figure 4, and assuming your Keycloak instance has been setup as an OpenID Connect Provider providing the authentication and authorization for your users. The first step is to fetch a web access token from Keycloak, this uses a CURL [10] command where you authenticate to Keycloak. Next, you pass the token using the `AssumeRoleWithWebIdentity` API to the Ceph Rados Gateway, this will return your ACCESS_KEY, SECRET_KEY, and Session Token. 3rd you simply add the returned credentials to your AWS client or Python script using the boto3 SDK and you can start using S3.

5. IO Performance Tests

We conducted write performance tests on two active systems: the Ceph S3 and our MinIO GW cluster, aiming to gauge and contrast the efficiency of each cluster. For testing S3 protocol performance we used the tool `S3bench` [11]. `S3bench` provides many options such as chunk size, threads, multiple endpoints, and other useful benchmarking options. Our initial tests we chose 1 MB – 100 MB chunk sizes and scaled from 1 – 100 threads.

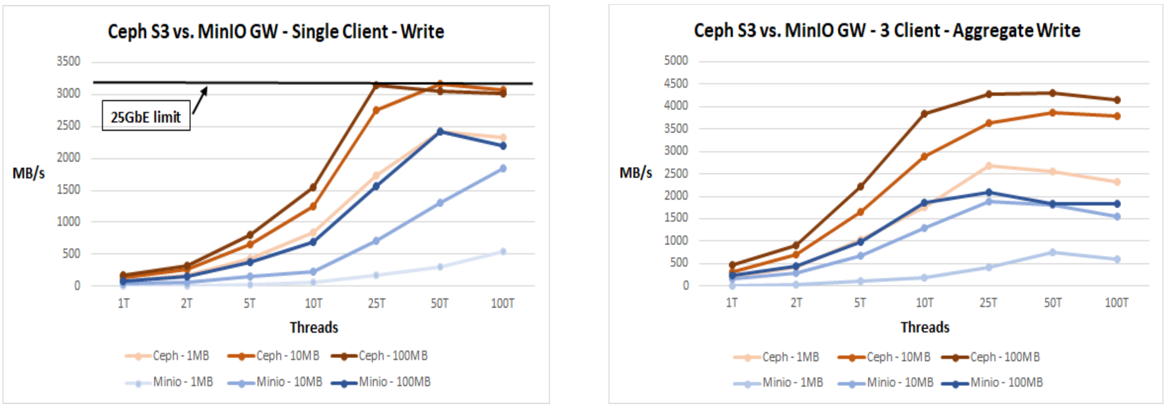
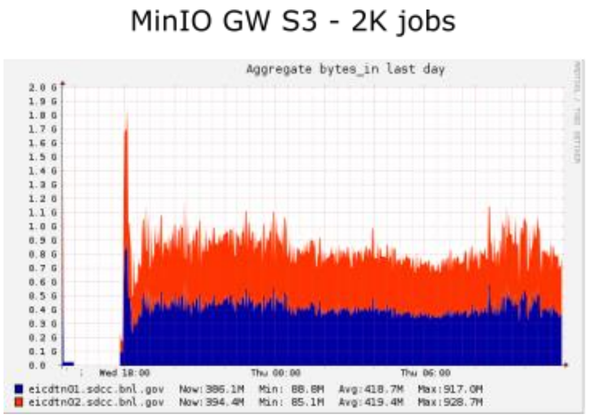


Figure 5. The graphs above illustrate the write performance test results for our Ceph S3 Storage cluster and the MinIO GW Storage cluster. The left-hand graph contrasts the write performance of a single client using different chunk sizes. Meanwhile, the graph on the right presents a comparison involving the combined write performance of three clients, again with varying chunk sizes.

As shown in Figure 5, the plot on the left illustrates single client write performance. The orange and brown curves are single client Ceph S3 write speed with chunk sizes 1 MB, 10 MB, and 100 MB. The blue curves are single MinIO GW S3 client write speed also with chunk sizes 1 MB, 10 MB, and 100 MB. Despite drive and size discrepancies, Ceph outperforms the MinIO setup among all chunk sizes. It is important to note that our backend Lustre storage for the MinIO GW is not using striping and may not be optimized for writes. Overall, a single Ceph client can saturate its outgoing 25 GbE link with intensive writing. For the plot on the right, each curve is the aggregate speed of 3 clients writing into Ceph or MinIO. The chunk sizes for both Ceph S3 and MinIO GW are still 1 MB, 10 MB, and 100 MB. The Ceph peak performance was 4.3 GB/s with 100 MB chunks and 25 threads. The MinIO peak performance was 2.1 GB/s with 100 MB chunks and 25 threads.



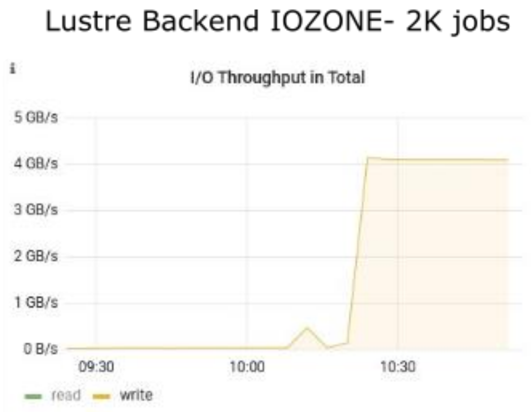
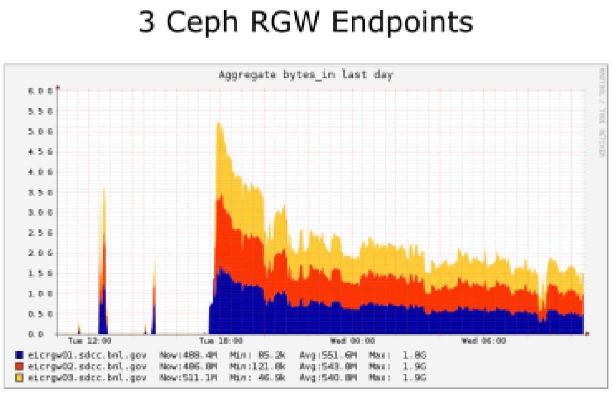


Figure 6. The left plot is showing the MinIO GW write performance during 2000 batch jobs each writing 10,000 x 10 MB chunks to the 2 MinIO GW endpoints. The plot on the right is the performance of 2000 batch jobs writing 10,000 x 10 MB IOZONE jobs to the backend Lustre storage that's used for MinIO GW

We wanted to consider the IO performance of the MinIO GW setup compared to the raw performance of its underlying Lustre storage. As shown in Figure 6, we submitted 2000 batch jobs using the `S3Bench` tool, each client writing 10,000 x 10 MB chunks into MinIO GW and another 2000 batch jobs using IOZone with each client writing 10,000 x 10 MB chunks per client. We found the MinIO GW peak write performance was ~50% (1.8 GB/s) compared to the underlying Lustre storage which reached 4 GB/s. Our MinIO GW setup has a considerable performance loss compared to its backend Lustre storage.



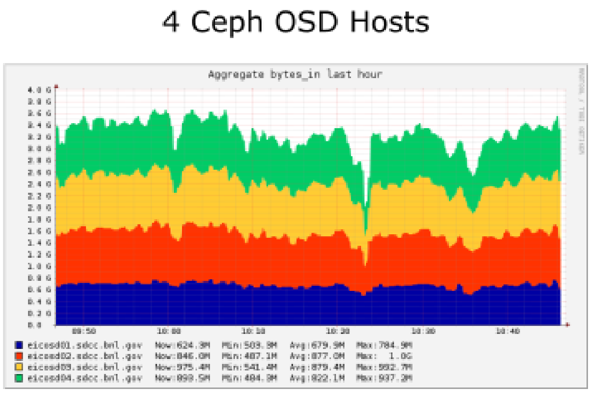


Figure 7. The left plot is showing the Ceph S3 write performance during 2000 batch jobs each writing 10,000 10 MB chunks to the 3 Ceph RGW Endpoints. The plot on the right is showing the aggregate throughput of the 4 Ceph OSD Storage hosts.

Our Ceph cluster currently consists of 31 Seagate 16 TB Exos x 16 hard drives. Seagate lists the manufacture spec. for maximum sustained transfer at 261 MB/s. Therefore at 31 disks (we had 32, 1 failed) the theoretical aggregate raw speed would be 8.1 GB/s (261 MB x 31). Instead of using standard replication in Ceph, which stores multiple copies of object files, we chose to use Erasure Coding (EC) [8]. With Erasure Coding data is broken into fragments of two kinds: data blocks and parity blocks and overall saves space relative to replication. As a starting point we used erasure coding 4+2 which is 4 data blocks and 2 parity blocks. With EC 4+2 the theoretical performance is 66% of the raw speed. As shown in Figure 7, on the left plot when submitting 2000 `S3bench` jobs wiring 10K x 10 MB chunks per host to Ceph S3 we reached a ~5.3 GB/s peak performance (roughly 66% of the Raw Speed w/ EC 4+2). The performance of the Ceph cluster is inline with what was expected.

Furthermore, Ceph efficiently distributes IO across all disks in the cluster. During our tests, a disk malfunction occurred, allowing us to delve into and verify the IO at the granular disk level. The right-hand plot in Figure 7 depicts the collective throughput for each host. As indicated in the legend, the peak throughput stands at approximately 1 GB/s. However, one of the hosts plateaus at just 785 MB/s — this is the host associated with the malfunctioning disk, confirming the IO discrepancy. Our analysis revealed that the utmost throughput per disk reached roughly 181 MB/s with EC 4+2. An advantageous takeaway from this is that we now possess a benchmark metric. This baseline can be instrumental in calibrating and meeting any overarching IO needs, right down to the specifics of individual disks.

6. Perspective

A single Realm Multi-Site Ceph Object Storage provides a global object namespace and ensures unique object ID's across the cluster. One of the great benefits of using S3 is the use of the HTTP protocol and the ability to access from anywhere with no special client software or kernel modules. While our object storage is focused on Ceph, a full MinIO object storage implementation or other object storage with S3 could be tested as options. Ceph provides a familiar technology from past deployments and is a solid baseline.

As of now, while specific IO requirements haven't been determined, the predictable scalability of our current cluster is noteworthy. Simply optimizing our existing setup to incorporate more disks, expanding from 31 to 48, would enable us to achieve an approximate maximum throughput of 8.7 GB/s. Taking scalability a step further, by doubling our host count to 8 and fully utilizing all disk slots, we can ambitiously target a throughput ceiling of around 17.4 GB/s. Such predictability in IO equips us with a precise metric for scaling and budgetary considerations.

7. Conclusion

The setup of our MinIO GW S3 accessible storage for the EIC production workflow was straightforward, facilitating the availability of a globally reachable storage solution. Despite its lack of certain critical functionalities, MinIO GW successfully served its intended purpose in this scenario. Moving forward, the construction of a Federated S3 storage system stands as a promising strategy, not just for the EIC, but for any project requiring a scalable, globally accessible storage solution. Such a system could be locally or globally expanded through multi-site utilization. In this context, Ceph object storage emerges as a robust candidate to facilitate the deployment of a multi-site S3 storage system. Leveraging its wide acceptance and substantial community support can significantly benefit the endeavour. As we pave the path for the next phases, our immediate focus narrows down to a series of imperative tests, including the refinement of Federated access and scrutinizing the effectiveness of a bona fide multi-site Ceph cluster. Ultimately, our aspiration is to craft a foundational structure for a multi-site Ceph cluster integrated with Federated ID access, poised for deployment and scalable to meet future demands. This skeleton or framework would stand ready to be deployed and scaled according to necessity, laying a solid groundwork for what lies ahead.

8. Acknowledgements

This work was supported by the Office of Nuclear Physics within the U.S. Department of Energy's Office of Science.

References

[1] MinIO Gateway https://min.io/docs/minio/linux/index.html
[2] The Ceph Object Gateway https://docs.ceph.com/en/quincy/radosgw/
[3] Keepalived https://keepalived.readthedocs.io/en/latest/introduction.html
[4] HAProxy https://docs.haproxy.org/2.6/intro.html
[5] Beats - https://www.elastic.co/beats/
[6] Ceph https://docs.ceph.com/en/quincy/start/intro
[7] M. Poat, J. Lauret – "Achieving Cost/Performance Balance Ratio Using Tiered Storage Caching Techniques: A Case Study with CephFS", (2016). (CHEP 2016)
[8] Ceph Erasure Coding https://docs.ceph.com/en/latest/rados/operations/erasure-code
[9] Secure Token Service https://docs.aws.amazon.com/STS/latest/APIReference/welcome.html
[10] CURL https://curl.se/docs/manpage.html
[11] S3Bench https://github.com/igneous-systems/s3bench
[12] Virtual Router Redundancy Protocol https://www.cisco.com/c/en/us/support/docs/security/vpn-3000-series-concentrators/7210-vrrp.html