# The OSiRIS Project White Paper

**Shawn McKee[1], Benjeman Meekhof[2], Ezra Kissel[3], Martin Swany[3], Andrew Keen[4], Nicholas Rahme[4], Michael Thompson[5], Matthew Lessins[5], Michael Parks[6]**

[1] Physics Department, University of Michigan, Ann Arbor, USA
[2] Advanced Research Computing, University of Michigan, Ann Arbor, USA
[3] OPeN Programmable Networks, Indiana University, Bloomington, USA
[4] Institute for Cyber-Enabled Research, Michigan State University, East Lansing, USA
[5] Computing and Information Technology Department, Wayne State University, Detroit, USA
[6] Network Planning, Michigan State University, East Lansing, USA

E-mail: smckee@umich.edu

**Abstract.** OSiRIS provides a distributed, multi-institutional storage infrastructure that lets researchers write, manage, and share data from their own computing facility locations. Our goal is transparent, high-performance access to the same storage infrastructure from well-connected locations on any participating campus. The project includes network discovery, monitoring and management tools as well. OSiRIS will provide data sharing, archiving, security and life-cycle management implemented and maintained with a single distributed service.

## 1. Introduction

The OSiRIS project has successfully connected our campuses with a software defined networking and storage system that will allow the seamless sharing of large datasets. Three and one-half years into the project we have established an infrastructure to provision and manage project services, deployed 7 Petabytes of Ceph storage, developed new tools for virtual organization management, and have more than a dozen virtual organizations on our platform. We can offer organizations access via S3, Globus, shell/scp, NFSv4 campus gateways, and direct Ceph cluster client connections. The process of user onboarding and service credential retrieval is automated and based on existing credentials at user home organizations. Our network links are constantly monitored for performance by an automated testing mesh laying the groundwork for more advanced network control. The remainder of the paper will cover the technical details, decisions and lessons learned in deploying OSiRIS, including information on how other institutions could deploy their own OSiRIS.

## 2. The OSiRIS Project Vision

OSiRIS (Open Storage Research InfraStructure) is a collaboration of scientists, computer engineers and technicians, network and storage researchers and information science professionals from University of Michigan/ARC-TS (UM), Michigan State University/iCER (MSU), Wayne State University (WSU), and Indiana University (IU) (focusing on SDN and network topology).

We are one of 4 NSF "Campus Computing: Data, Networking, Innovation: Data Infrastructure Building Blocks" (CC*DNI DIBBs) projects funded in 2015. OSiRIS is

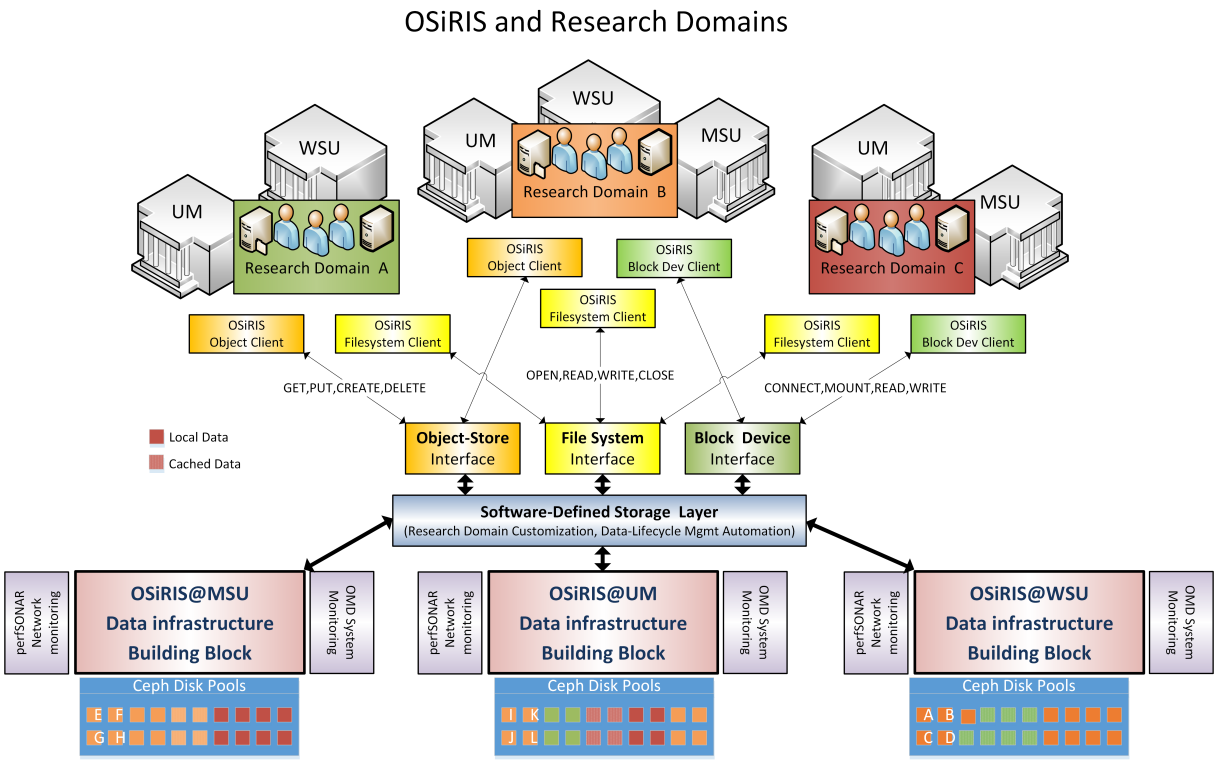


**Figure 1.** Logical Overview of OSiRIS

prototyping and evaluating a software-defined storage infrastructure, initially for our primary Michigan research universities, designed to support many science domains. Our goal is to provide transparent, high-performance access to the same storage infrastructure from well-connected locations on any of our campuses. By providing a single data infrastructure that supports computational access in-place, we can meet many of the data-intensive and collaboration challenges faced by our research communities and enable them to easily undertake research collaborations beyond the border of their own universities.

A single scalable infrastructure is easier to build and maintain than isolated campus data silos. Data sharing, archiving, security, and life-cycle management can all be implemented under one infrastructure. At the same time, our architecture will allow the configuration for each research domain to be optimized for performance and resiliency (Fig. 1).

## 3. OSiRIS Project Components

Services provided by OSiRIS rely on the coordination of Ceph object storage, client storage interfaces, networking, identity management, and provisioning resources. Our requisite core components are openly available but required significant integration or enhancement work to turn them into a complete platform linked to federated identities. In turn the work we have done is also openly available as will be noted by references in the following sections.

### *3.1. Ceph*

Ceph is a distributed object storage system that gives us a robust open source platform to host multi-institutional science data. The core of Ceph is the Reliable Autonomic Distributed Object Store. RADOS is self healing, self managing replication, and has excellent scalability and performance[1]. RADOS supports multiple data interfaces including POSIX, S3 compatible

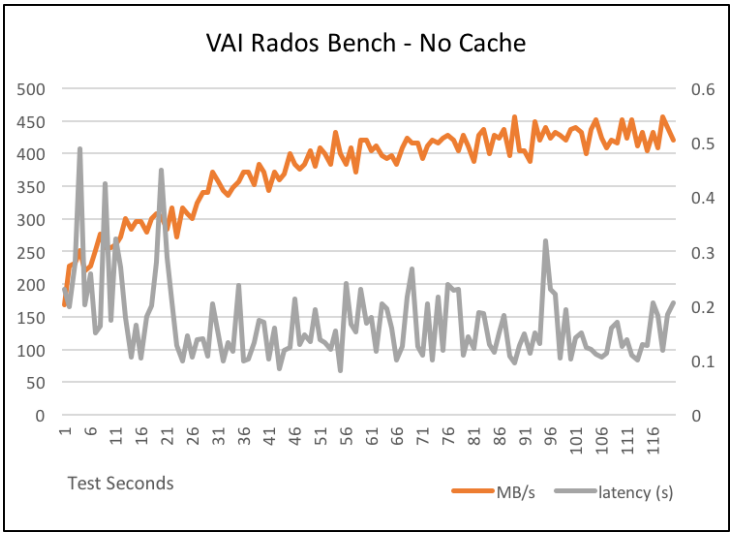


**Figure 2.** Ceph client benchmark of write throughput from Van Andel without cache overlay shows speed limited by network from VAI to other OSiRIS sites

object storage, and kernel block devices. Ceph has sophisticated allocation mapping using the Controlled Replication Under Scalable Hashing (CRUSH) algorithm to allow us to customize data placement by use case and available resources[2].

Our Ceph deployment is distributed across sites at WSU, MSU, and UM. Ceph allows us to choose the level of replication among these sites based on the needs of participating science domains. Typically our highest level of data resiliency would be provided by having one or more replicas at each site. Ceph also has options for creating Erasure Coded data pools which provide configurable redundancy similar to RAID.

*3.1.1. Ceph Cache Tiering* Ceph allows for overlaying storage pools with a 'cache tier' pool which can be composed of entirely different storage elements[3]. The intended use-case is to overlay slower storage with a small/fast storage pool to cache frequently used data. Our project found that we can also use cache tiering as a local overlay to pools composed of geographically spread out elements.

We initially experimented with this usage at the Supercomputing 2016 conference in Utah[4] before deploying it for active use. Our first use-case was the Van Andel Institute in Grand Rapids. We deployed 3 storage nodes to this site hosting fast NVMe storage and configured our Ceph CRUSH map to place the replicas for certain pools only on these nodes(Fig. 4).

Users at Van Andel access storage via an NFSv4 (Ganesha) service co-hosted as a container on one of the three storage nodes. For data sets which fit within the cache capacity ( 11 TiB) the NFS server can read/write entirely to the cache pool. Once the pool has been filled, or enough time has passed, Ceph automatically will begin flushing data to the backing pool located at the other OSiRIS sites. Cache flushing triggers are configurable but to begin with we did not change the default values which start flushing dirty cache objects at 30% full and clear more space by flushing clean objects (hot data for reading) at 70% full.

Benchmarks performed against this arrangement show significant increases in write throughput and corresponding decreases in operation latency when using a pool overlaid by the local cache (Fig. 2 and Fig. 3 )

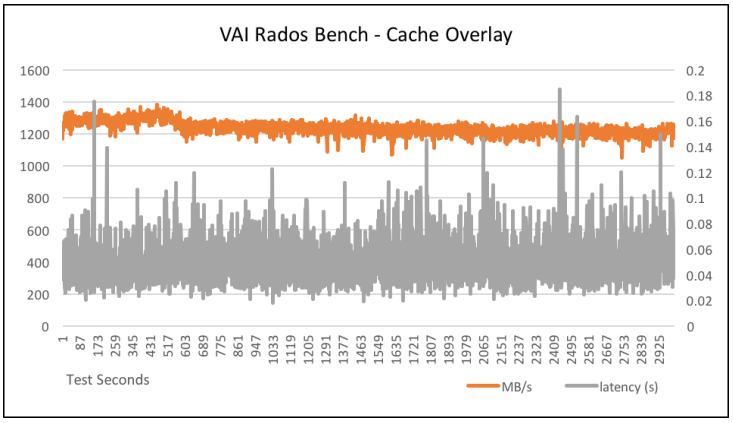


**Figure 3.** Ceph client benchmark of write throughput from Van Andel with a local cache overlay shows dramatic reduction in latency and increased throughput due to IO using cache storage

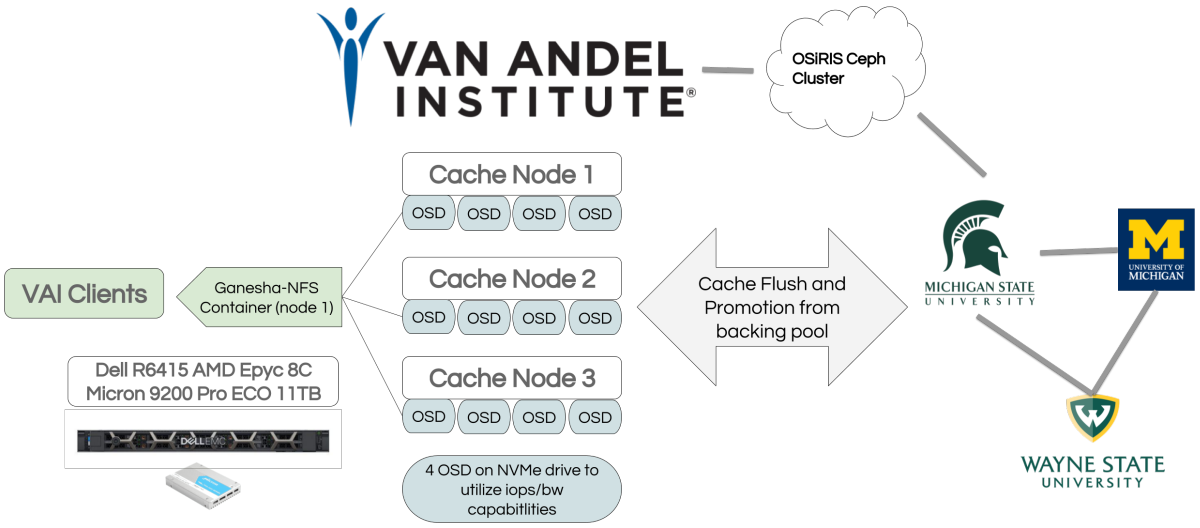


**Figure 4.** Van Andel Ceph cache tier deployment

*3.1.2. Ceph CRUSH Primary OSD* The Ceph CRUSH map also gives us the option to choose 'primary OSD' within any given set of replicas. This primary OSD is used for client reads and writes. Any given data pool will have many data 'placement groups' (PG) each of which will have a different set of OSD so clients ultimately end up reading from and writing to a large set of primary OSD.

A typical Ceph CRUSH rule will only broadly define placement group replication buckets. For example, the default Ceph replicated CRUSH rule only specifies that data replicas must end up on different hosts. It doesn't specify which hosts those are. Likewise, our OSiRIS default replication rule says that data replicas must end up at different sites without necessarily specifying an order (there are only three sites and three replicas, so one constant is that every site is used). In either of these examples the primary OSD in placement group replica sets will end up being randomly distributed across the replica hosts or sites.

CRUSH rules can also be more explicitly defined. Such a rule might dictate that the first

replica goes to one site, the second to another, and so on. The order of the replica distribution is important because the first OSD chosen for a replica always ends up being the primary in a set. If we were to say that WSU is the first replica in our CRUSH rule then all the primary OSD will be at WSU. When clients request a read or write operation they will be directed to the primary OSDs. For those clients close to the primary OSD, this means that their data read operations will not have to get data from OSD outside of the WSU site. Write operations will still require replication to offsite OSD. In our tests this optimization can significantly boost read throughput and iops for clients if they are near the primary OSD[5].

Another advantage of this optimization is that it can be easily adjusted to some other location with a copy of the data without causing data movement. Changing the crush rule to prefer a different primary OSD simply changes the classification of the OSD involved.

*3.1.3. Ceph Network Considerations* As a multisite distributed deployment OSiRIS faces some unique network challenges. In particular we do not necessarily have equal network bandwidth between all of our sites. Given two sites linked by a very fast link which are connected to a third site by some fraction of that faster link, the smaller link will be completely saturated by data replication once enough OSD become involved. For example, bringing up a new host at the low-bandwidth site will cause a large amount of replication (backfill) traffic and degrade cluster operations as the link saturates. In our case the solution was to adjust Ceph OSD settings to scale back these recovery operations. The settings we found most relevant were osd_recovery_max_active, osd_backfill_scan_min, osd_backfill_scan_max, and osd_recovery_sleep_hybrid. Recovery sleep has the most pronounced effect[6].

Ceph performance is also sensitive to network latency. As previously noted there are optimizations we can apply to avoid network latency for some clients. Within the context of OSiRIS, the 2-3 ms latency between our sites is not a significant obstacle to functionality, but we expect that we cannot achieve certain performance levels with pools replicated across three sites. It would not be reasonable to compare the performance for those pools to an equivalent number/type of Ceph nodes deployed in a single data center, but we are nonetheless able to meet the expectations of our scientific users. OSiRIS also has the option of allocating pool replicas to a specific site instead of across three sites, and in that case latency between sites is not a factor for most operations. We have had experience distributing the deployment to sites geographically farther away (Utah[4] and Texas[7]) with a slight loss of performance but retaining functionality.

Clustered resources like Ceph do not respond well to certain kinds of network disruptions in a WAN/multi-site deployment. In a three-point network 'loop', losing one leg of connectivity without an alternate route puts Ceph services into an unstable state. One Ceph OSD might report OSD at another site down while the third site is still reporting it up. This effect cascades into severe service degradation as OSD are alternately marked up or down and the cluster attempts to repair the data replicas. Likewise cluster management services see this effect on a smaller scale. There is no easy solution to the issue. The best mitigation is redundant network paths, perhaps actively managed as is being researched in our NMAL work (see section 3.4). Second level mitigation is active monitoring of network connectivity followed by temporarily setting 'noout' and/or 'nodown' cluster settings that prevent OSD from falsely being reported down. It may also be required to shutdown services at one of the three sites to avoid conflicting status responses for cluster management services.

*3.1.4. Ceph Metrics* Ceph is a complex system with a variety of metrics for monitoring performance and resource usage. OSiRIS in particular needs to obtain resource usage grouped by virtual organization, which in our case maps to Ceph data pools.

Initially the project used a ceph plugin for the Collectd utility[8] feeding metric data to InfluxDB. The plugin collects data directly from Ceph daemon 'admin sockets' in a JSON

format. It worked and continues to be a viable option but had some disadvantages: A tendency to break when new Ceph releases come out, collecting large amounts of less-relevant data from the socket, and missing some information we were interested in about pool usage. This led us to look into an alternate solution based on the new-at-the-time ceph-mgr daemon plugin architecture. The mgr daemon as a data source was particularly appealing because it collects and makes available metrics from all Ceph daemons in a central cluster service. This seemed advantageous over running a Collectd daemon to collect it from individual admin sockets, or at least advantageous over separately collecting data that already is being collected.

The first version of the ceph-mgr InfluxDB plugin was written by a student at U-M working for our project[9]. We found and continue to find it useful for collecting essential Ceph cluster metrics including pool data usage, daemon metrics, pool placement group status, and more. The work was contributed to the Ceph project under the student's name and has seen interest and further activity since then. At a later date she also contributed code to include pool placement group information.

Fig. 5 shows an example of our cluster overview dashboard and Fig. 6 is an example of pool usage stats. We also have dashboards collecting Ceph RGW (S3) operations, MDS (CephFS) operations, and detailed OSD statistics such as replication latency. There are many metrics exported via the mgr plugin which cover most insight one would want into the cluster. The scope is purposely limited to more useful stats by definitions in the Ceph code assigning priority to each metric.

For new deployments we also recommend looking into other time-series databases for collecting metrics. Prometheus has a similar ceph-mgr plugin included with Ceph and can be written to by Collectd for other metrics. Graphite does not have a ceph-mgr plugin at this time but is supported by Collectd. Additional ceph-mgr plugins include Telegraf, Zabbix, and potentially more with each release[10].

### *3.2. Storage Access*

It was previously noted that a Ceph cluster supports multiple data access methods. Most usage of Ceph does not involve users interacting directly with it as an object store but instead connecting services that use Ceph underneath to store data objects and meta-data. For example, CephFS requires a metadata server (MDS) to handle POSIX filesystem semantics and this MDS itself uses the Ceph cluster to store filesystem structural data such as inodes and permissions. Ceph S3 (RGW) operates similarly as an implementation of the Amazon S3 protocol using Ceph RADOS as backend object storage and also for storage of service operational metadata such as users, bucket indexes, etc. There are some uses of OSiRIS that connect directly to our Ceph cluster as well. The following sub-sections will cover the details of how we have deployed storage protocols with Ceph.

*3.2.1. NFSv4 and CephFS* A critical feature for providing network mountable POSIX storage outside of a controlled environment is a means to verify identity and map that identity to POSIX numeric uid or some other means of access restriction. CephFS itself does not provide this capability and determines permissions based on the numeric POSIX information as provided by the client. It is, however, available using NFSv4 id mapping in combination with Kerberos credentials.

We use the umich_ldap id mapping backend to look up Kerberos principal attributes defined in NFSv4RemotePerson LDAP objects in our directory[11]. The key attribute linking to an OSiRIS user is their Kerberos principal in their institution realm. We do not actually operate our own Kerberos realm. Through standard Kerberos mechanisms the Ganesha NFS server verifies the user identity in a realm they already belong to and looks that up in LDAP to link it to an OSiRIS POSIX identity. A conceptual overview is provided in Fig. 7. Setting this up


**Figure 5.** Example of OSiRIS Ceph cluster dashboard using stats collected by ceph-mgr InfluxDB plugin. The information presented here can tell us about cluster activity as well as cluster problems such as excessive latency, down OSD, or high load on systems


**Figure 6.** Example of OSiRIS Ceph cluster pool dashboard using stats collected by ceph-mgr InfluxDB plugin. The information on this dashboard tells us who is using OSiRIS at a given moment, their cumulative data usage, and whether users are getting expected performance.

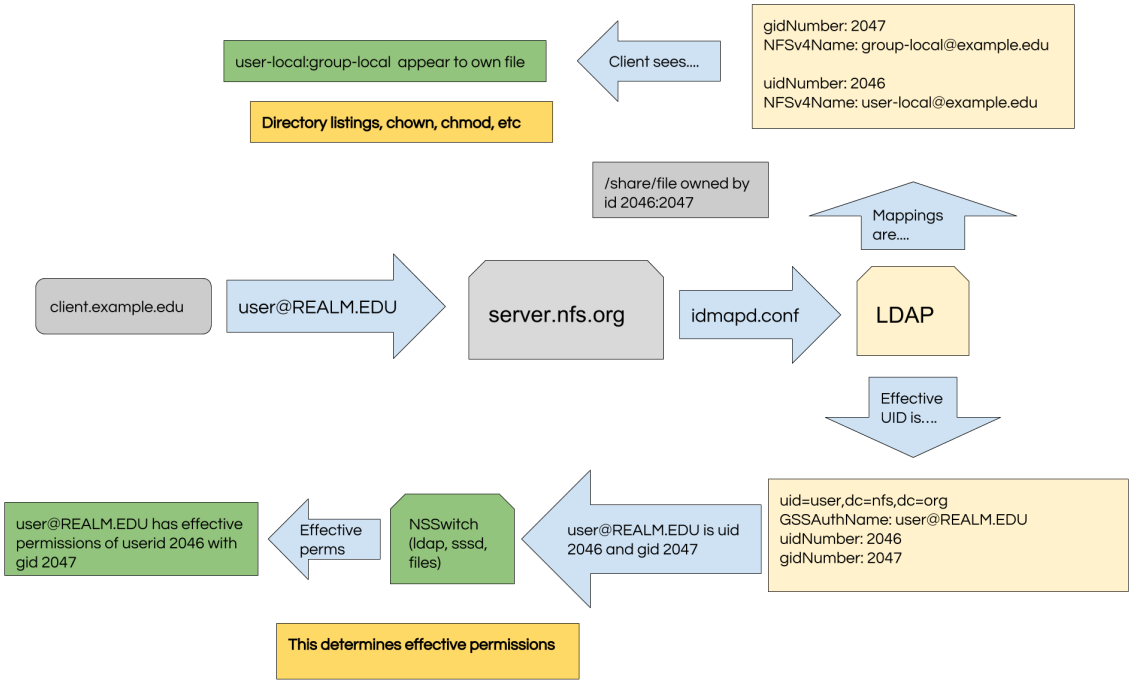


**Figure 7.** NFSv4 identity mapping for CephFS with Ganesha. The figure shows the sequence of looking up user KRB5 principal in LDAP to map to OSiRIS uid (bottom half) as well as the mapping of OSiRIS uid/gid to local users and groups (top half)

requires some coordination with campus admins to obtain Kerberos keytabs for our server and for any clients which need to mount the NFS export. This is a fairly standard request and has not been an issue at any of the OSiRIS campuses.

It is an eventual goal that users be able to configure their own Kerberos principal mappings to OSiRIS identities with COmanage inserting their choice into the LDAP directory. Additionally they should be able to configure local groups to map to OSiRIS groups. As a first step we have instead chosen to write a script to manage these mappings. Any OSiRIS admin can use the script to modify or view idmap settings for a user and it includes command-line documentation. The script is available from our LDAP utility repository[12]. Given that Kerberos and idmapping can be a confusing subject, it may be best to require interaction with OSiRIS admins to configure. For campus settings we could potentially script these mappings by using the Edu Person Principal Name (eppn) that is typically going to match Kerberos realm principal for the user.

*3.2.2. HAproxy and Ceph RGW* Ceph RGW is a storage service which provides an S3 compatible interface. Originally specified and provided by Amazon, the 'Simple Storage Service' is based on http[13]. Ceph's implementation of the service is compatible with most aspects of the API that are not specific to Amazon services. There is a large ecosystem of S3 compatible clients and programming libraries, and it is well suited for working with data at a large scale across many locations. S3 eschews traditional filesystem concepts such as directories and POSIX permissions and instead organizes data into 'buckets' which hold objects. Complex ACL can be assigned to buckets and objects and there are also fairly simple predefined ACL such as authenticated-read, public-read, etc. Many tools can present an S3 listing in more traditional terms where buckets appear as directories and objects as files in those directories.

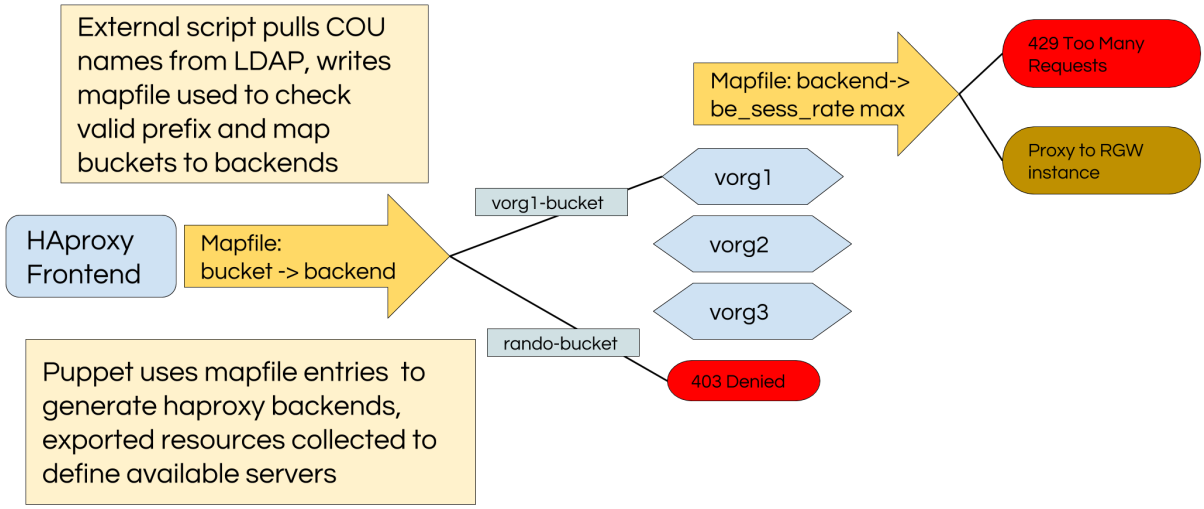


**Figure 8.** Using HAproxy to route request and limit request rate by VO. Requests are mapped to virtual org backends based on a bucket prefix and each backend can have a maximum session rate associated via external dynamic mapfile.

In our implementation of Ceph RGW services, we chose to put RGW instances behind HAproxy instances which are configured to load balance requests across multiple backends. Each OSiRIS site has at least one RGW + HAproxy instance, perhaps more. Requests are load balanced across instances at all sites to maximally leverage our available infrastructure. We can also tag HAproxy instances in our Puppet configuration to group them together and keep traffic confined to backends in that group. This is useful for sites like Van Andel or other future edge sites that are best served by only utilizing backends that are in their LAN.

To generate the backend configurations we first generate a map file by querying our LDAP directory for groups matching a certain pattern used by COmanage for the COU it creates (CO_COU_Name). We then winnow these down into a set of unique names and write a map file for HAproxy to link bucket name to backend name. The script also creates several static mappings for legacy buckets and services such as the admin REST API and Swift. The additional mappings are important because we also will be matching incoming bucket requests against this list to allow or disallow. The backend map file is read into a Puppet custom fact which generates an array of backend names for our puppet manifest to iterate and create the HAproxy backend configurations.

HAproxy can parse incoming request headers, modify headers, and make decisions as to how to process the HTTP request. We use this functionality to group requests into specific backends named after the science virtual organizations in OSiRIS. Our HAproxy rules verify that buckets are named with a prefix that matches a known virtual org and then route the request to the appropriate backend matching that virtual org. Additional rules in the virtual organization backends check the rate of incoming requests against an external map file and if it becomes too high the backend returns a 429 error (Too Many Requests). Map files can be controlled outside the HAproxy configuration and modified without restart by using the HAproxy socket interface[14]. Currently we do not dynamically manage the rate map file but instead set a rate that prevents any one org from completely saturating our capabilities (4000 sessions per second). Fig. 8 shows the decision flow from client request to HAproxy backend choice and rate comparison.

*3.2.3. dCache and Ceph Rados Block Device* Storage sites in the ATLAS project often use dCache to provide storage. Typically the backend storage for dCache nodes is traditional attached storage. Since the advent of Ceph the dCache project has also added the capability to use Ceph RBD as dCache storage pools[15]. The OSiRIS project shares a PI with the ATLAS Great Lakes Tier 2 site on the U–M campus which has facilitated experiments using this storage method. At the time of this writing work is underway to compare Ceph backed dCache pools in replicated and erasure–coded configurations.

*3.2.4. Globus* Globus provides a worldwide accessible data transfer service with web-based user interface. OSiRIS operates a total of six Globus endpoints, two per storage location. One is sufficient but Globus does not currently allow combining storage back ends so we must operate one for each available back end.

Each site hosts a Globus endpoint connected to our CephFS storage and another endpoint connected to our S3 (RGW) storage backend. The endpoints are managed under the U-M Globus license which gives us Globus share capabilities. A Globus share is configured by users and allows sharing a specific path in Globus with other Globus users. The share accesses the underlying storage using the access rights of the user that created it and that user is responsible for allowing access to other Globus users or groups.[16]

Globus has several methods for authenticating users but we use only the CIlogon method[17]. When activating a Globus endpoint the user is directed to cilogon.org where they authenticate with their institutional credentials. The Globus server then receives a certificate DN as user identity. Our servers are configured to take that DN and map it to a local OSiRIS user. Currently Globus uses a map file to do this. In our configuration the map file is kept on a shared (CephFS) filesystem and synchronized with the LDAP attribute VOPersonCertificateDN[18] associated with each OSiRIS user. Multiple certificate attributes can be assigned. Our documentation directs users to send their certificate DN to our help email and any OSiRIS admin can then feed that DN to a simple script[12] which inserts it into the LDAP directory and refreshes the map file.

With some development work a new gridmap module could be written to directly query LDAP[19]. We may take this project on depending on available resources. Managing the user certificate DN(s) for mapping would be improved by using COmanage to allow them to input their own instead of requesting an OSiRIS admin to do it. Obviously there are scaling issues with having an administrative person be involved. COmanage includes a 'CertificateAuthenticator' plugin which may help to enable this feature or provide a code template for adding the feature.

*3.3. Networking for OSiRIS*

A central component of our distributed OSiRIS Ceph deployment is the network that interconnects our sites. Because OSiRIS is designed to be a multi-institutional infrastructure, this means we need to integrate the networking required to deliver an effective OSiRIS with the policies and networking infrastructure in place at each University. This becomes even more challenging when you consider that OSiRIS, as a Dev-Ops infrastructure, strives to mix production quality services with the ability to develop cutting-edge capabilities on leading edge technologies. This is often in conflict with the requirements of institutional networks who must maintain a rock-solid network for their users.

The network modifications and additions for OSiRIS started during the first few months of the project. The critical first step was in identifying the right people to participate in an OSiRIS networking group from each institution and ensuring that these people had adequate time available through direct or indirect funding. We found it very important to identify network engineers with practical experience who were also very aware of their institutions policies and

best practices. Our goal was to find at least two network / IT engineers from each institution, at least one of whom could participate in regular technical and planning meetings.

Having identified a network team, we began with an initial meeting with them, the OSiRIS project engineers and the OSiRIS NMAL team to discuss the OSiRIS vision and the unique networking environment at each institution. This initial meeting allowed us to share information about each institution and to discuss how the OSiRIS equipment would be connected at each University and inter-connected between Universities. One advantage for OSiRIS was that our Universities shared a common fiber infrastructure, jointly owned and operated by all three that allowed us to initially interconnect MSU and UM at 2x40Gbps and and WSU to both MSU and UM at 10 Gbps. Of course, having the fiber infrastructure is only part of the challenge. We needed to get connections and configurations in place to allow it to work. One of the primary lessons learned was that identifying the required subnets needed at each institution is one of the first things that needs to be done.

For OSiRIS, we realized that Ceph, because of its self-healing characteristics, could use a "back-end" network to support inter-site Ceph traffic as data is replicated to fix failed hardware. The would keep the science users on a "front-end" network and could help minimize contention between Ceph operations and science data flows. However, even two networks are not optimal because we need a separate management network to control our servers, IPMI cards and network devices. We would recommend that any group planning to deploy OSiRIS elsewhere, start by planning out three networks at each institution that will participate: front-end, back-end and management.

OSiRIS was designed from the start with Software Defined Networking (SDN) as an enabling component. For those who have tried to implement SDN solutions before, there are three challenges to address: 1) hardware capabilities (or lack thereof) in the network devices, 2) lack of production quality software implementations and 3) network engineer's lack of familiarity with SDN capabilities, requirements and best practices. As noted above, the initial efforts were focused on basic network design, configuration and deployment. However the longer term goal was in creating a networking infrastructure the would allow OSiRIS to orchestrate traffic flows to both avoid congestion between network users (Ceph operations, science users) and to maximize the capacity between resources by utilizing multiple distinct paths. This goal implies an infrastructure that is very different from the principles used to deliver a robust, production-quality network and this tension is one of the interesting aspects of trying to create a dev-ops infrastructure within a production network infrastructure. For OSiRIS to make progress, we realized we needed to work closely with each campus, accounting for existing policies and determining how to safely take "risky" steps in ways the minimize that risk.

We learned a number of things about how to make progress while minimizing risks. Potentially disruptive network changes had to be carefully analyzed, and reasonable mitigation's had to be developed before those changes could be enabled. For example, in configuring our Dell Z9100's for OpenFlow use, we needed to reconfigure the network to fail-safe if something happened to the OpenFlow instance or its associated controller. The need for a management network, distinct from the front and back end networks was part of this learning experience. One beneficial result of working together was raising awareness and interest in SDN capabilities and requirements.

One of the interesting aspects of OSiRIS was the fact that two of the three institutions hosting our distributed Ceph deployment were interconnect at 80 Gbps while the third was only connected at 10 Gbps. This initial configuration was forced on us by the existing connectivity already in place and it presented a interesting opportunity to understand the impact of having asymmetric site connectivity between Ceph locations. What we learned was that during Ceph self-healing, the connection to the less well connected site would become saturated, severely impairing end users of data at that site as well as impacting other users of that network

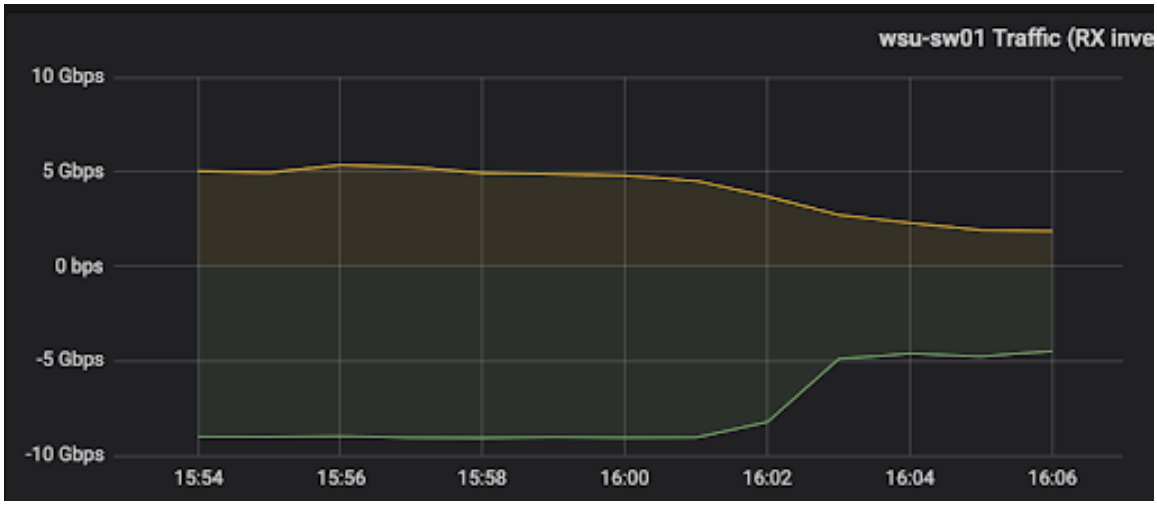


**Figure 9.** Shown is the network traffic at WSU during Ceph recovery operations before and after a Ceph reconfiguration to limit the network traffic on the sites 10 Gbps link.

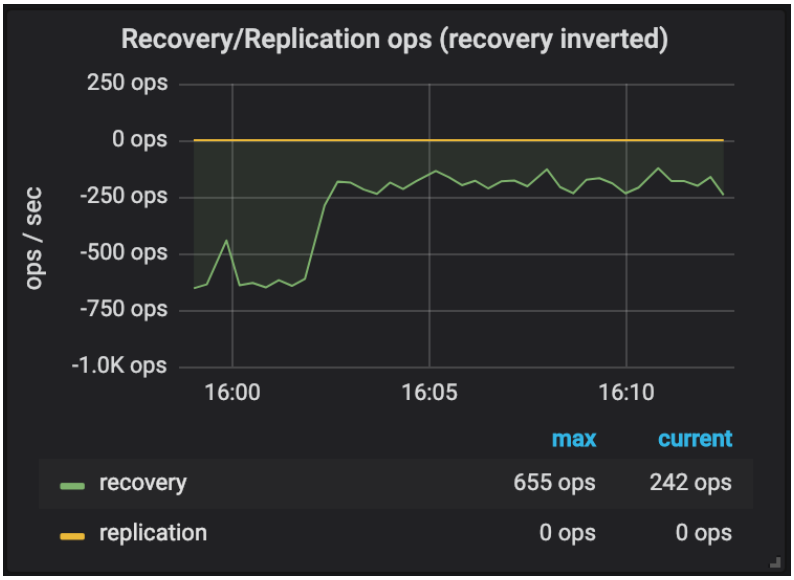


**Figure 10.** Shown is the number of Ceph recovery operations before and after a Ceph reconfiguration to limit those operations.

connection. Shown in figures 9 and 10 are the results from tweaking the following Ceph variables:

```
osd_recovery_max_active:      1        # (default 3)
osd_backfill_scan_min:        8        # (def 64)
osd_backfill_scan_max:       64        # (def 512)
osd_recovery_sleep_hybrid:  0.1        # (def .025)
```

You can see that lowering recovery tuning fixes the issue, at the expense of under-utilizing our other faster links. Recovery sleep had the most effect, the others impact is not as clear. The longer term solution to this type of problem will be to utilize network orchestration to either move the traffic over other paths or limit the maximum bandwidth these flows to a reasonable fraction of the available bandwidth.

*3.4. NMAL (Network Management Abstraction Layer)*
Another important part of the OSiRIS project is active network monitoring, management and network orchestration via the NMAL. Network topology and perfSONAR extended with Periscope monitoring components deployed to hosts and switches ensure that our distributed system can optimize the network for performance and resiliency through SDN (Software Defined Networking) control of components.

Main components in NMAL include Periscope (UNIS, Measurement Store, and Runtime System), Flange Network Orchestration Service (NOS), and an SDN controller.

- UNIS - The Unified Network Information Service (UNIS) combines the capabilities of resource lookup and network topology services within a general data model and reference implementation. Accessible via a RESTful API, UNIS maintains descriptions of multi-layer topologies and associated measurement metadata along with the services running within as embedded objects, allowing UNIS to answer complex queries about the network and its current operational state.
- BLiPP is a flexible framework for collecting host metrics for reporting metadatas back to the network information store (UNIS). The BLiPP framework supports the model of a generalized measurement point: any allowed host command may be executed to gather a set of metrics and the parsed values are normalized into timeseries data and sent to one or more Periscope measuement stores.
- Flange NOS - The Flange domain specific language (DSL) allows for driving network configuration and management with declarative network topology assertions. The Flange NOS uses information from UNIS to apply user-defined network behavior given the constraints of the underlying known topology. Flange in NMAL provides a Ryu SDN controller interface for manipulating paths in the network.
- SDN Controller - Driven by information collected in UNIS and managed by Flange, an SDN controller can dynamically modify network topologies to enable the best path between clients and data and between internal OSiRIS components (i.e., for Ceph replication).

*3.4.1. Topology Gathering and Monitoring* Realizing the network abstraction and management goals of NMAL requires the ability to keep an accurate and up-to-date representation of our connected resources. NMAL's discovery tools are built around managing metadata in UNIS so we can maintain a standardized mechanism for reasoning about the network. Our primary means for Layer 2 network discovery is the use of SDN controllers which listen to traffic from SDN-capable switches that update UNIS in response to network changes. We also incorporate tools built around SNMP and Traceroute to supply other types of data to UNIS such as the service capabilities, PerfSONAR regular performance testing, and other Layer 3 metrics.

With the representation of the network developed by our discovery tools we are able to abstract network monitoring and supply a centralized GUI for network administrators to diagnose the current state of the network at a glance. The monitoring interface displays all discovered resources in UNIS along with the corresponding metadata such as from regular performance testing (Fig. 11). A strength of NMAL's approach to discovery and monitoring is that the metadata we collect to provide an abstracted view of the network can also be used to reason about and apply dynamic changes in response to network events.

*3.4.2. Coordinating SDN flows* The Flange NOS utilizes an up-to-date picture of the network generated by topology discovery and measurement tools in order to provide automatic provisioning services to network administrators with minimal direct configuration. OSiRIS administrators provide Flange with a program describing the parameters to be enforced on the network. These Flange programs describe the behavior of the network at the flow level.

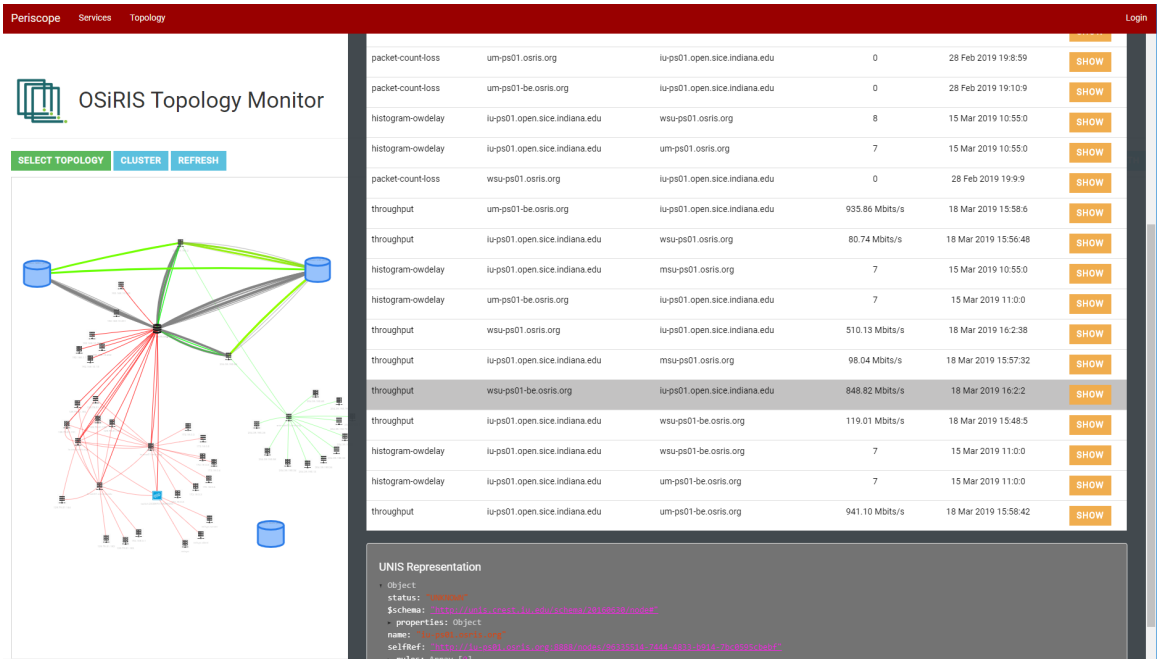


**Figure 11.** NMAL Topology Viewer and Regular Testing Results Overlay

Each instruction defines - either permissively or prohibitively - how data may flow between endpoints. In the OSiRIS context, this allows administrators fine control over the source of data, how the data is routed between sites, and which sites have connectivity at the network level.

Flange aggregates network topological and measurement information in order to solve for the best paths through the network for each asserted data flow. Critically, this is not a static analysis; if conditions on the network change, Flange adjusts the maintained flows accordingly, routing around congestion or outages. Each flow is inserted into the network using a configurable back end module. For OSiRIS, we chose to use the Ryu controller which provides switch configuration through a restful API which the Flange NOS configures as conditions change on the network.

*3.4.3. Testbed Demonstrations* We have demonstrated the use of NMAL orchestration capabilities in collaboration with SLATE (Services Layer at the Edge), controlling an Internet2 SDN testbed using SLATE-hosted resources to run the Flange NOS, UNIS topology service, and perfSONAR network measurement containers. The Flange NOS software was used to control dynamic paths on a slice of the Internet2 SDN testbed along with resources provided by the SLATE platform. This testbed provided an evaluation opportunity for the developing NMAL components outside of the production OSiRIS storage network. We were able to demonstrate traffic flows being diverted dynamically between two different paths based on active measurement feedback. PerfSONAR measurement test points were used to provide active monitoring of each segment of the path controlled by Flange. A Flange declarative program was used to control the selection of traffic flow between SDN-enabled switches based on a predetermined measurement threshold applied on either a throughput or latency dimension (Fig. 12). With this program, when the link becomes congested, we observed a drop in performance on our test flow across the designated circuit. After a short detection delay the performance returned to nominal levels as the flow was routed around the congested link. We have found that for small graph systems - such as those represented by OSiRIS - the performance of Flange NOS is dominated by network

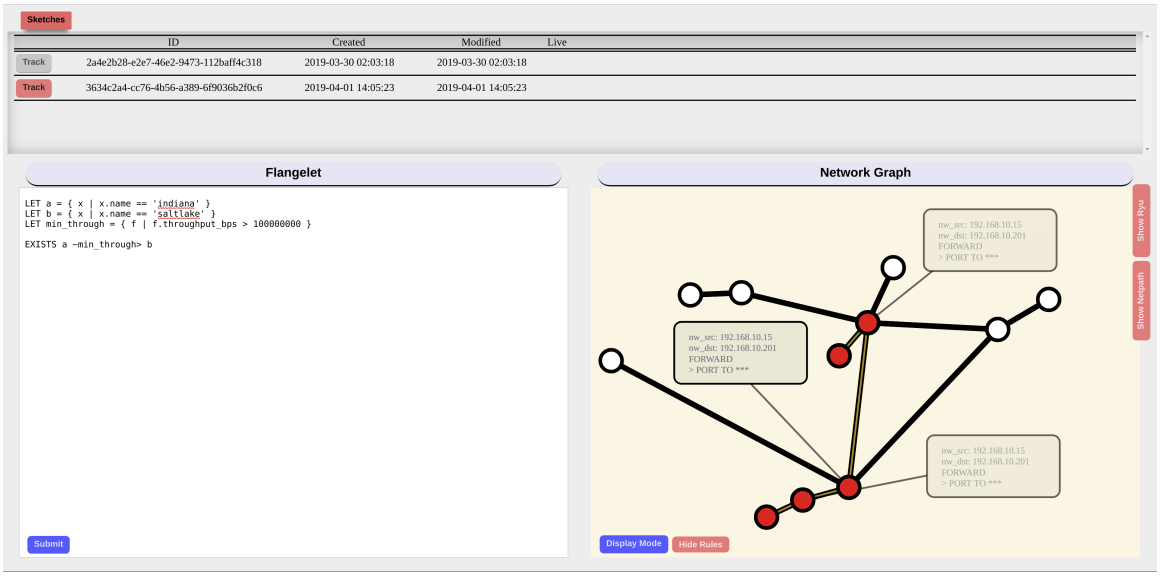


**Figure 12.** The Flange NOS generating flows from measurement feedback in the Internet2 SDN testbed.

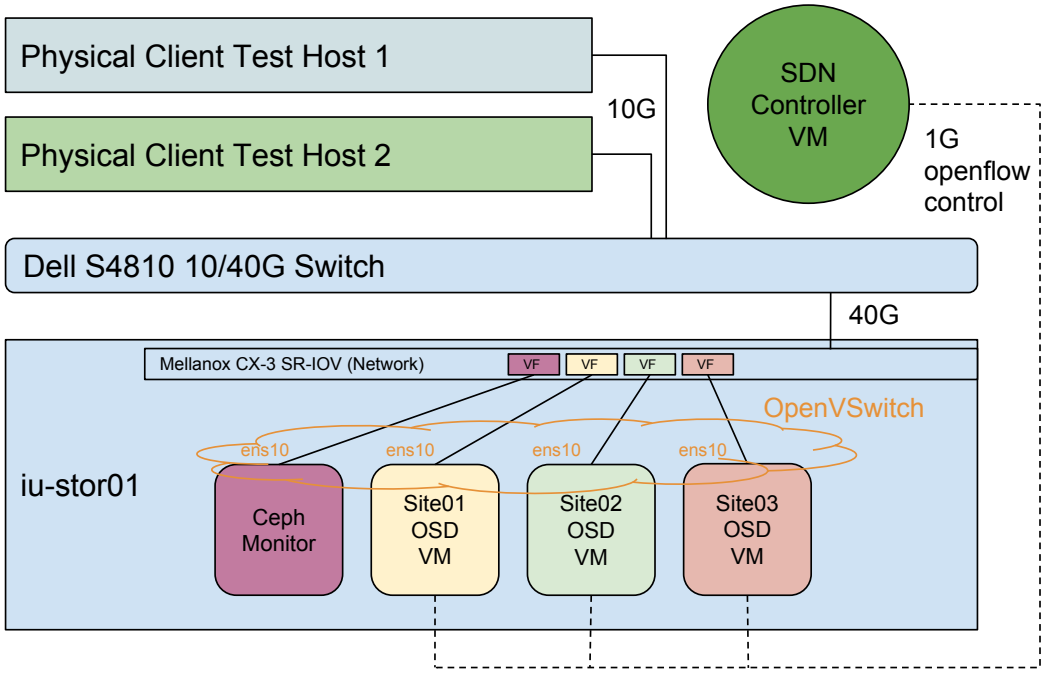


**Figure 13.** Testbed at IU to evaluate QoS tuning of Ceph cluster network traffic

measurement resolution. Internal network analysis and state modifications take a fraction of the time compared to the smallest recommended interval between measurement values.

*3.4.4. NMAL Network QOS* With a mature implementation of UNIS and Flange, one of the overarching goals of NMAL is to better manage the flow of traffic between OSiRIS sites. Such flow management is of particular importance when there exists asymmetric bandwidth between

one or more deployments or if there is simply a site that has less network capacity, which presents a bottleneck condition during normal Ceph operations such as replication and backfill. This fact has necessitated throttling of Ceph operations to avoid creating denial-of-service behavior at sites with less bandwidth than other deployments.

One approach to resolving the capacity imbalance is to enable quality-of-service (QoS) configuration at each site to manage traffic originating from Ceph operations at the network layer. We are evaluating two general mechanisms to address the issues described above:

(i) Apply priority queues to ensure that adequate bandwidth exists for Ceph client operations to prevent timeouts and delayed read/write performance.

(ii) Apply traffic shaping to provide better transport protocol performance between sites with asymmetric link capacities. This is of particular importance when latency between sites is increased.

We have developed a testbed at IU to develop and evaluate the priority queues and traffic shaping techniques using a software-based SDN controller called OpenVSwitch (OVS) (Fig. 13). Each OSD VM in the testbed topology represents an OSiRIS site to which we can apply various latency, bottleneck, and QoS paramaters to using a combination of OVS and Linux traffic classification. The advantage of using OVS is that we can apply the QoS configuration dynamically using our existing Flange approach, which interacts with the Ryu SDN controller managing the OVS instances. This testbed provides an opportunity to stress-test the mechanisms being evaluated, observe the behavior on a controlled Ceph installation (emulating the software running on OSiRIS), and develop a set of best practices for eventual deployment at the production sites.

### *3.5. Authentication, Authorization and Accounting*

The OSiRIS approach to authentication is to use identity federations and avoid managing authentication accounts. Federation participants will use their local providers to verify an identity and begin a self-service enrollment process which creates an OSiRIS identity belonging to one or more OSiRIS virtual organizations. We leverage InCommon[20] and eduGAIN[21] federations, Shibboleth[22], and COmanage [23] to enable our science domain users to self-enroll, self-organize, and control access to their own storage within OSiRIS.

Virtual organizations are known as 'COU' internally to COmanage (CO organizational units). Once a user is approved by designated VO admins or OSiRIS project admins provision their identity is established in COmanage and linked to their institutional identity. From there access to OSiRIS services is provisioned as shown in Fig. 14. Access to service credentials is via the COmanage gateway. Should an individual move organizations we can simply link their new organization to existing OSiRIS identity. Multiple federated identities can be linked to a single OSiRIS identity as well.

Virtual OSiRIS organizations can self-organize and manage members and roles via OSiRIS services such as COmanage and Grouper. They can further control access to data via service specific tools such as S3 ACL, Globus shares, or Posix permission tools.

*3.5.1. COmanage LDAP and Grouper Provisioners* Provisioning user identities to LDAP is a core feature of COmanage. Users who enroll into OSiRIS are pushed out to our LDAP directory with various attributes defined in COmnage mapped to LDAP object attributes. These include posix attributes, user-managed SSH keys for shell access, email, and edu Person Principal Name (ePPN) from their institution. We can then use these identities as needed to link the user to any system with LDAP capabilities.

One such system is Internet2 Grouper[24]. The core function of Grouper is to pull in identities (subjects) from a variety of sources and use them in defining roles and groups. COmanage

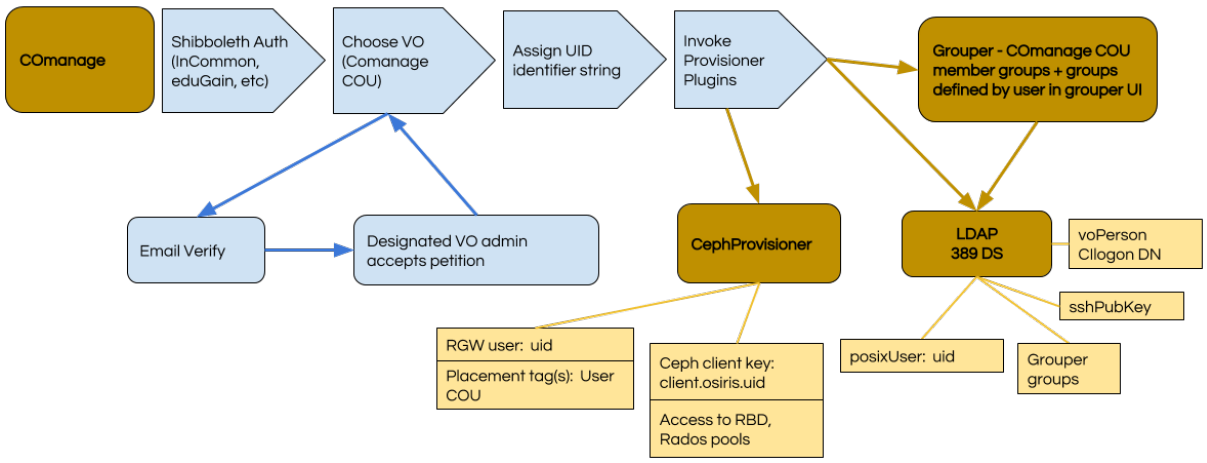


**Figure 14.** COmanage flow for new user authenticating to our COmanage gateway and joining a virtual organization. Following verification and approval the COmanage provisioning plugins translate identity and access information as appropriate for services including LDAP, Grouper, and Ceph

includes a plugin to push information to Grouper so that COU (virtual org) memberships defined in COmanage become groups in grouper.

COmanage defines three unchangeable groups automatically for every virtual org: active members, all members, and admins. These become groups in Grouper with members as relevant. Additional groups can also be defined in Grouper using the identities previously established by COmanage or by defininging groups of groups. Grouper, similar to COmanage, pushes group membership information to LDAP so we can use it as needed. Any complexities of maintaining group of group memberships are handled internally by Grouper so the net result is LDAP groups with appropriate memberships.

The virtual organization itself becomes a 'stem' or folder in the Grouper hierarchy and we can assign permissions to this stem allowing virtual organization admins the capability to create their own groups and assign memberships to them. The ability to manage new groups is limited to their own virtual organization folder. An administrator for one virtual org could not alter groups for another virtual org. An example screenshot showing the Grouper hierarchy is shown in Fig. 15.

We use the information in LDAP throughout the project. CephFS integrates as any filesystem would with POSIX users and groups from LDAP. Users in virtual organizations automatically belong to auto-created groups from COmanage and we can use those groups to set reasonable default permissions on filesystems directories shared by virtual organizations. Additional groups created in Grouper are also available for use in assigning file ownership and capabilities.

*3.5.2. COmanage Ceph Provisioner Plugin* COmanage has no built-in capability to provision users to Ceph, but it is designed with a plugin-based architecture. Different identity management events in COmanage trigger calls to configured plugins with information about the event. Events might include new user identifier, new user groups (new COU membership), request to reprovision user, and more. OSiRIS created a new plugin under this architecture to handle provisioning storage for virtual orgs (, link storage pools to CephFS directories or S3 placement targets, create users, and assign user capabilities .

The plugin is freely available from our Github repository.[25] Using and installing the plugin follows the normal COmanage plugin installation instructions available from their

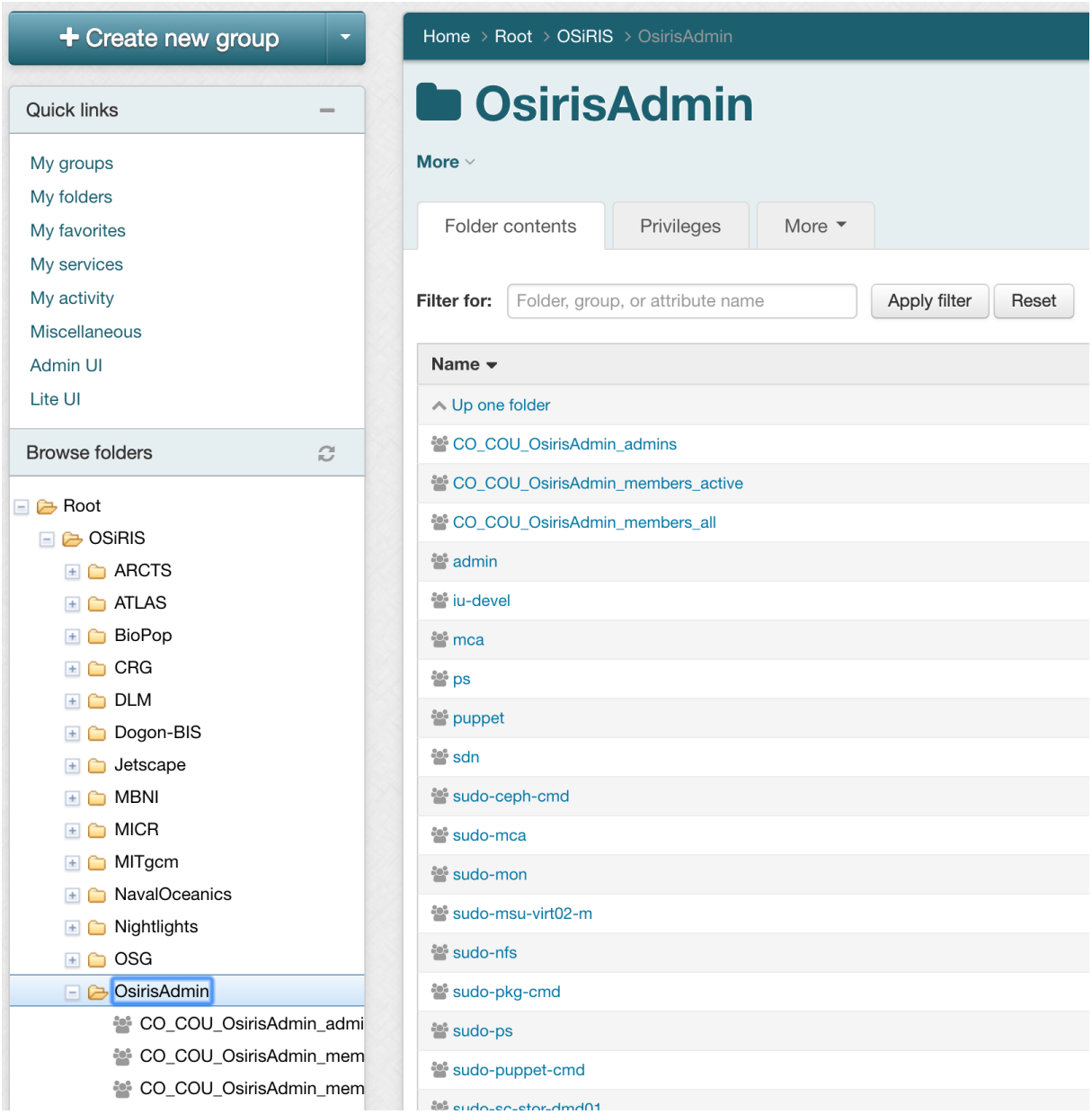


**Figure 15.** Grouper tool for managing group memberships by virtual org admins. Virtual orgs such as our own OsirisAdmin group for project admins become stems (folders) which contain groups which can inherit folder permissions.

documentation.[23] More detailed instructions specific to our plugin are available on our website, and these documents also cover in more detail the structure of the plugin described in Fig. 16 and Fig. 17).

*3.5.3. SSSD* Internally our project leverages users and groups from LDAP read by the System Security Services Daemon[26] to control system access and privileges:

- SSSD provides a utility to lookup SSH public keys from LDAP for ssh shell access. COmanage allows users to self-manage their keys. OSiRIS admins and science domains

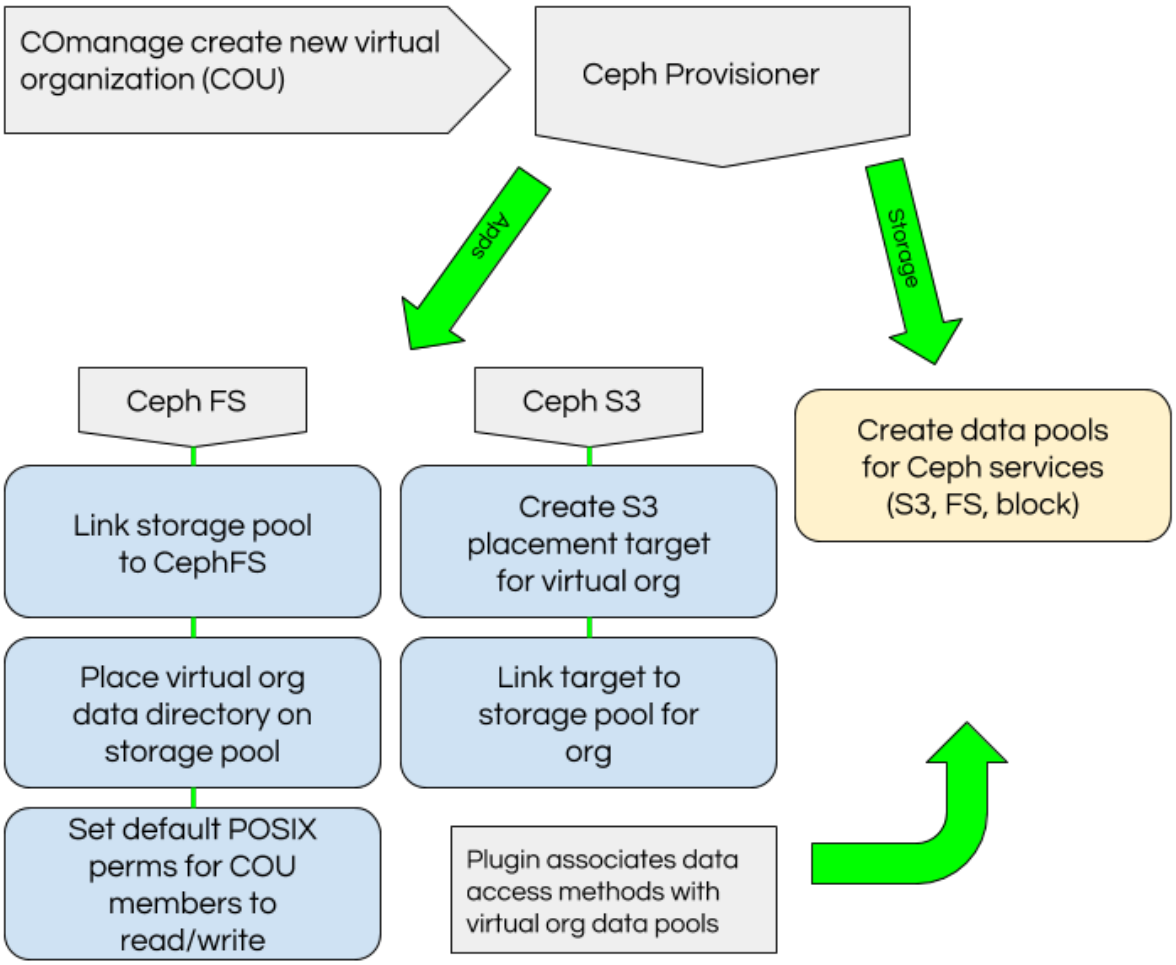


**Figure 16.** COmanage provisioning steps for Ceph VO storage include new pools for CephFS, S3, Rados, and steps to link those pools to directories or S3 placement locations

use the same functionality

- Login to systems can be restricted by LDAP group. Our puppet configuration manager generates configuration dynamically allowing users in certain groups matching the host 'role' to login as well as users in certain broad administrative groups
- Sudo privileges can be controlled by LDAP groups. We have a scheme whereby groups in Grouper named with 'sudorole' or 'sudohostname' map to LDAP groups used to allow sudo commands on hosts. LDAP sudoRole objects are populated for each of our hosts by using Puppet 'exported resources' which define LDIF files for import into our LDAP directory when Puppet collects those resources on an LDAP directory host.

*3.5.4. LDAP Directory* An LDAP directory is core to organizing OSiRIS identities and access to services. This key piece of our infrastructure is deployed in a multi-master replicated arrangement with a replica at each of our main sites. There is no specific implementation of LDAP required for any OSiRIS component but our experience has been positive with the '389'[27] directory server. This open source LDAP server is also the basis for Redhat Directory Server. Deployment and replication setup is managed by a Puppet module which we wrote

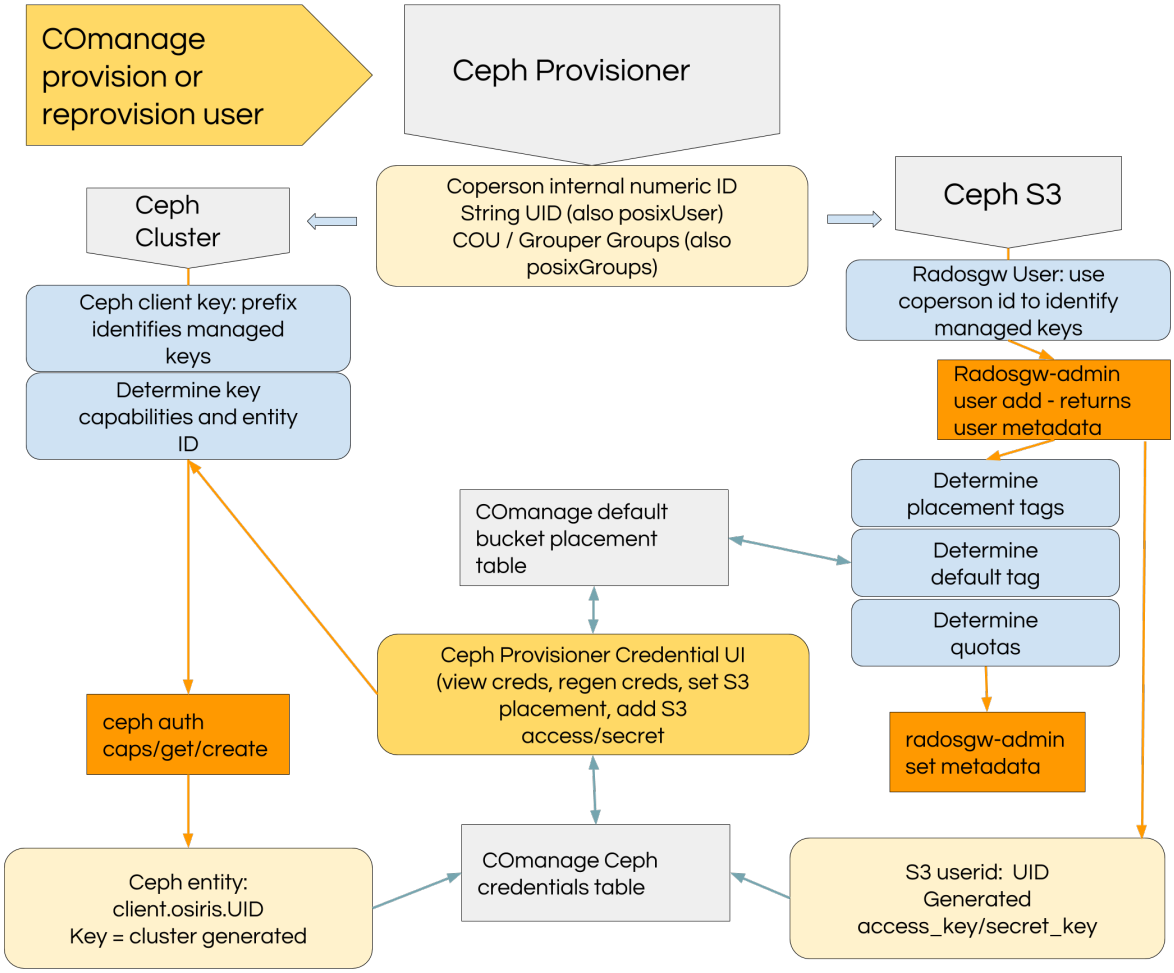


**Figure 17.** Flow diagram shows relations between COmanage and Ceph information for virtual organization users. Provisioning is idempotent and users can be re-provisioned to verify or update as needed.

for the purpose[28]. It is also possible to configure replication using CLI instructions or a management GUI[29].

Some features depend on LDAP schema not included with the directory server. They are all included in the previously noted Puppet module and also available elsewhere.

- NFSv4RemoteUser: Available from our puppet module source tree or from the original documentation[11]. Specific use of this schema is covered in section 3.2.1
- voPerson: Schema defining a variety of useful attributes for managing people in virtual organizations[18]. At this time we use the attribute voPersonCertificateDN to store CIlogon certificate DN for use by Globus identity mapping as described in section 3.2.4.
- ldapPublicKey: Schema defining an attribute to store SSH public key. Available in openssh-ldap package available in any RHEL derived OS or easily found via web search. We include a copy of the schema in our Puppet module.

*3.6. Data Lifecycle Management in OSiRIS*

While not a primary driver of the OSiRIS infrastructure, data lifecycle management (DLM) has been of strong interest to our team, both because of the additional value it could provide to our science domains and the attractiveness of utilizing some of the software defined storage capabilities in Ceph to automate aspects of DLM.

Our project proposal integrated Information and Library Science experts at Michigan State University, University of Michigan and Wayne State University to explore what might be possible in the DLM area. In conceptualizing what DLM functionalities might be useful to grow and support in the OSiRIS environment, our library collaborators expressed an interest in researching data curation. Data curation can be defined as the activity of managing and promoting the use of data from their point of creation to ensure that they are fit for contemporary purpose and available for discovery and reuse over time and by other persons outside of the original research team. In particular, they were interested in working with us to explore a curation in place model. Rather than moving data from a space were it is actively being developed into a space where it would be curated, the curation in place model proposes that the data not be moved and instead curation and preservation functionalities be built into the active work space. Reducing or eliminating migration would help prevent the loss of data and contextual information that often occurs as a result of moving the data from one environment to another.

The first step in their research was conducting a survey designed to explore what functionalities a curation in place system might have based on the priorities and needs of the researchers who are using OSiRIS currently. The results indicate that respondents saw providing support for metadata and documentation as the most important functionalities of data curation. Although respondents saw generating metadata and documentation as an important task, many of them were only somewhat satisfied with the results of their work. There was a particular interest in developing functionalities in the OSiRIS infrastructure that would reduce the labor and effort required in generating robust documentation and metadata. The next step taken by our librarian collaborators was to inventory some of the data sets currently hosted within OSiRIS to understand how the data was currently structured and organized, as well as to identify the nature and depth of any accompanying documentation or metadata. The data sets they reviewed did not include much beyond a very high-level description of their contents, indicating that there may be a need to develop tools, resources and guidance in the OSiRIS infrastructure to generate and capture metadata to enable curation functionalities. We envision future work with our library collaborators centering around discussions about applying what was learned about desired functionalities from their survey and comparing them to existing curation functionalities within the OSiRIS infrastructure to determine where best to invest developing capabilities for data curation.

## 4. OSiRIS Deployment

Deployment and management of OSiRIS resources leverages existing open-source technologies and collaborative workflows. As diagrammed in Fig. 18 configuration changes are versioned through Git. Any authorized OSiRIS admin can create a branch to test new changes before submitting them as a 'pull request' for integration into our production configuration. Infrastructure services are hosted as virtual machines on KVM/Libvirt systems. These are built on Foreman, which is lifecycle management tool for physical and virtual servers which integrates closely with Puppet. Foreman also has capabilities to integrate with other configuration tools such as Ansible or Salt.[30]

*4.1. Foreman and Puppet*

Foreman is structured with a remote 'smart proxy' architecture allowing for a small proxy build server to be controlled at any remote site by one central instance. It is also capable of

provisioning new VMs and can thus be used as a complete integrated solution to configure and deploy new VM instances at any of the OSiRIS sites (Fig. 19).

Foreman also has functionality to create offline 'bootdisks' through the foreman-bootdisk plugin[31]. We used this functionality to deploy OSiRIS storage servers to the Grand Rapids based Van Andel institute. Using a Foreman created boot image the machines there were built to our typical template and integrated into our configuration management similarly to the existing sites.

The well-known Puppet tool was our choice for configuration management in OSiRIS. Puppet's yaml-based hiera database for configuration data simplifies storing parameters specific to site, node type, node role, etc. In fact, any arbitrary hierarchy based on any arbitrary node data can be used to organize parameters used for configuration. Puppet also supports a tool called 'r10k' which is designed to automatically map Git branches as Puppet configuration environments. Using this tool and shared Git repository, we have a workflow that enables project admins to collaborate on configuration enhancements and changes without disrupting the stability of our production configuration. The r10k/git/puppet workflow is a common best practice[32]. Puppet integrates tightly with Foreman so that we can use the Foreman GUI to choose which Puppet git environment a node will use for configuration.

### *4.2. Docker*

OSiRIS has also leveraged containerization for some services. Our configuration for NFSv4 as described in section 3.2.1 has been baked into a container which is documented and available for reuse or extension on the public Docker hub[33]. The container is focused on the configuration we use with LDAP idmap look up and Kerberos authentication but also designed to be flexible enough for others to use in their environment. Container specifications are easily forked and modified as needed for those wishing to use it as a starting point.

NMAL services described in section 3.4 are also available as containerized deployments on Docker hub[34]. Containers available include the UNIS, Flange, DLT and Topology service. Usage of these containers is described in the README files included with each container.

In our context we use a Puppet module to manage installing Docker and running the containers on hosts providing NFS or the various NMAL services. Containers are deployed as microservices on dedicated VMs as opposed to using a container orchestration service such as Kubernetes. The initial driver for the containerization work was our collaboration with the SLATE project which does leverage Kubernetes to utilize the NMAL containers[35].

## 5. Science Domain Engagements

OSiRIS aims to engage with science domains in many different fields which benefit from a shared data storage platform. Domains so far include ATLAS, Naval Oceanography, Genomics at UM Institute for Social Research, Global Nightlights satellite datasets, Microscopy, Imaging and Cytometry Resources at WSU, the JETSCAPE project at WSU, and others. We are also in planning phases to provide storage to the nationwide Open Storage Network and IceCube neutrino detector. The following subsections highlight a few specific science domains of interest.

### *5.1. ATLAS*

ATLAS compute jobs use the OSiRIS Ceph S3 gateway to read/write single events. By reading only a single event at a time, ATLAS can leverage transient computing resources to run short jobs as available. ATLAS has been successfully using this service with OSiRIS for about two years at this point. A high-level overview is provided in Fig. 20.

The ATLAS ES workflow also involves 'merge jobs' that fetch output from ES jobs at the CERN object store. Recently ATLAS approached us looking to improve the workflow for transferring merge job input by using Globus to move this data onto the OSiRIS object S3

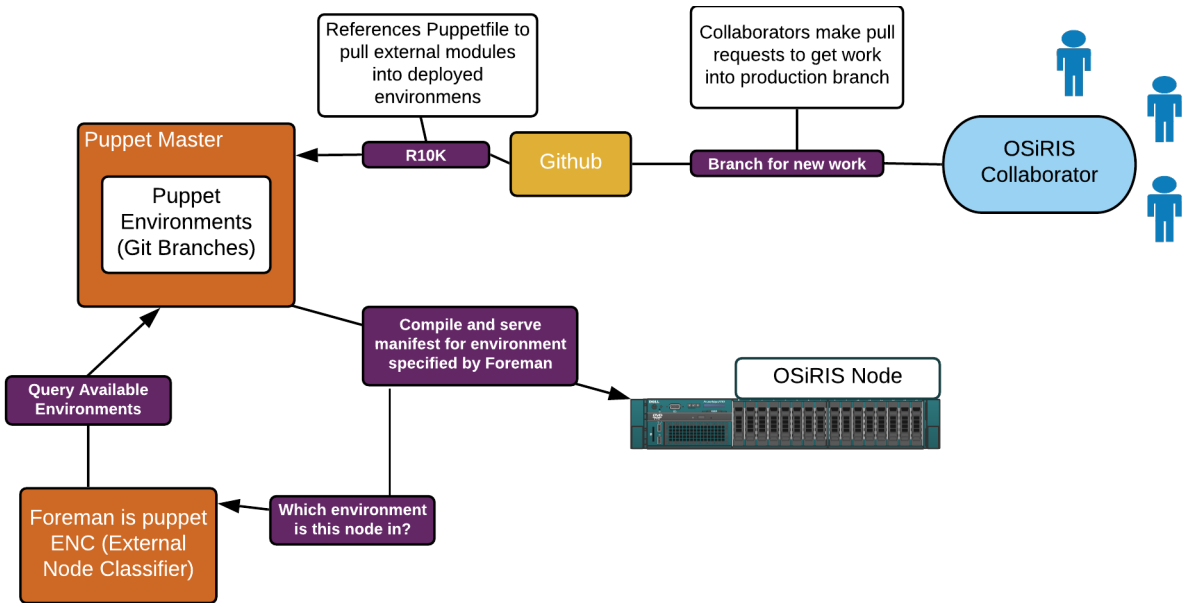


**Figure 18.** Puppet environments are created from Github branches by the r10k tool. OSiRIS admin collaborators can develop and merge changes in independent branches applied to one or more nodes controllable in Foreman.

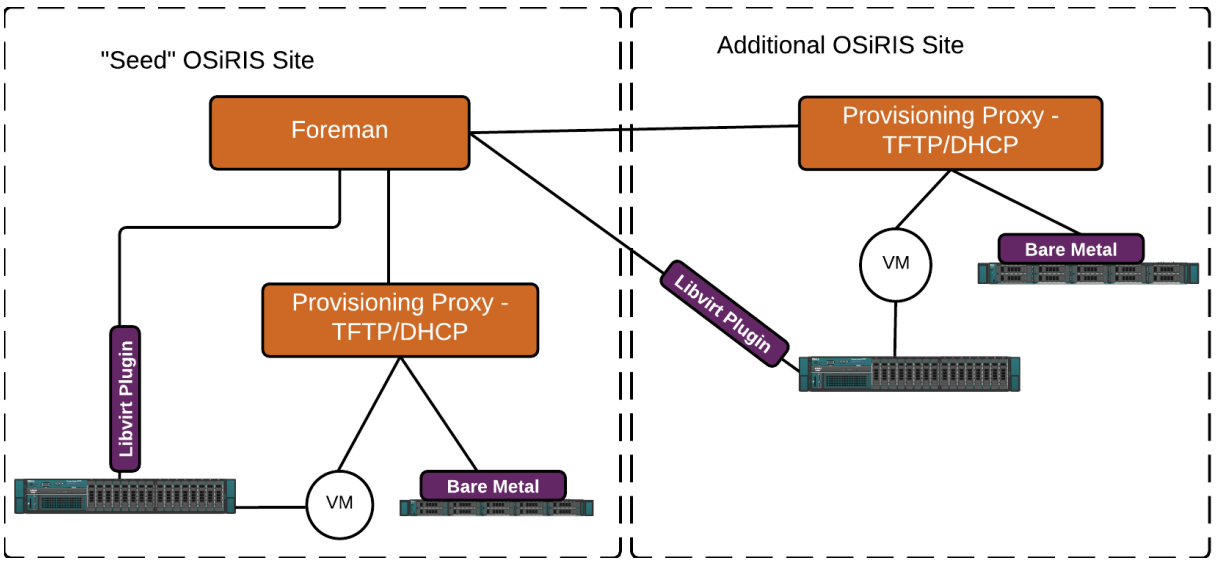


**Figure 19.** Foreman smart-proxies and libvirt plugin allow us to deploy VM or bare metal resources to remote sites controlled by a central instance.

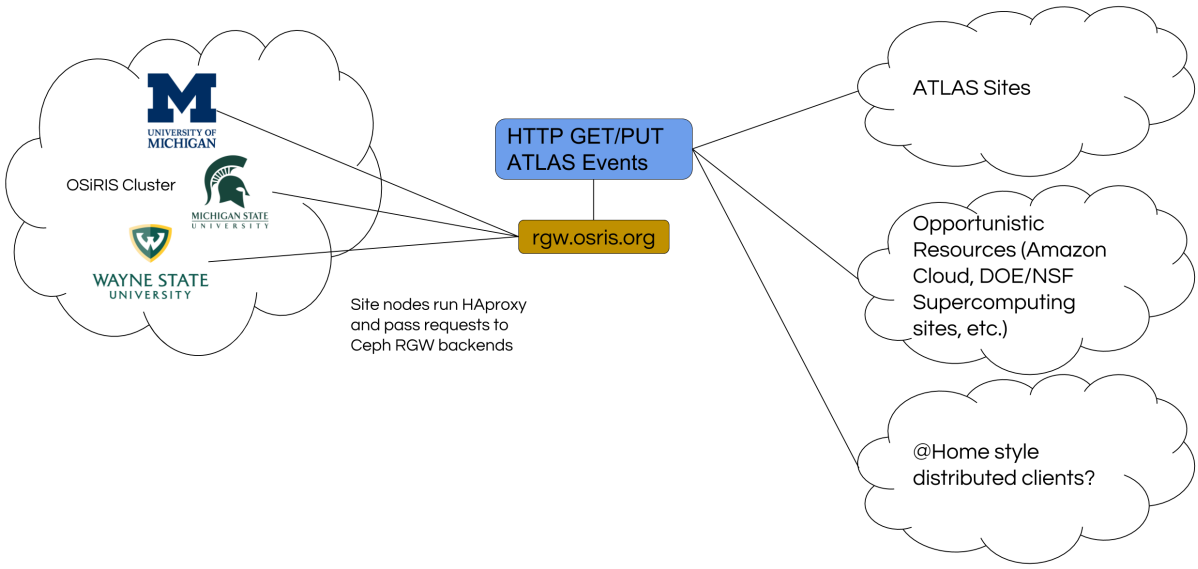


**Figure 20.** ATLAS Object Store architecture allows many types of compute resources to retrieve input for jobs from OSiRIS RGW. A single RGW URL spreads workload across all our member sites.

object store. The details of how this was done previously are outside the scope of this paper but the proposal was that OSIRIS stand up Globus S3 connectors to be used in the data transfer flow. As it happens we were already in process of doing this as a general improvement and these new endpoints are now online. The success of this approach is unknown at the time of this writing but so far it looks promising.

There are other known methods for accessing ATLAS resources in Ceph. The Dynafed project[36] from CERN includes plugins to access S3 endpoints such as ours via the ATLAS Dynafed namespace. The GridFTP plugin which was created and is used by the RAL ATLAS Tier-1[37] is also a potential interface. We have not explored these interfaces but they remain as potential options.

*5.2. Naval Oceanography*

The US Naval Research Lab is collaborating with researchers at UM to share their high-resolution ocean models with the broader community by storing them with OSiRIS. We currently provide space via CephFS and accessed through our transfer gateways via scp, FDT, and Globus. We have seen users from academic institutions and contracted commercial collaborators enroll into the Naval VO to use the data. The researchers themselves have had positive comments for OSiRIS: "OSiRIS is a very useful tool for us, because it allows us to share the outputs of model simulations run on Navy and NASA machines, with scientists who do not have accounts on those machines. OSiRIS therefore allows for a wider dissemination of model information."[38]

*5.3. Biology Population Studies (Dogon-BIS)*

Researchers in this virtual org perform expression analysis using RNAseq on an Illumina platform. In order to distinguish maternal from paternal transcripts they have to generate sufficient sequences at positions where the two transcripts differ. This requires deep sequencing resulting in large files ($\tilde{1}$7 Gb sequence file). The mapping of these sequences to the genome generates BAM files that are also large ($\tilde{7}$ Gb). It is computationally expensive to generate BAM files so they archive the files on OSiRIS for re-analysis with more up to date workflows.

There is also interest by population geneticists to use this data to determine genetic affinities with other ethnic groups to solve the question of their origin and time of isolation. As such the group uses the OSiRIS platform to share the data with population geneticists at the University of Michigan[39].

*5.4. Van Andel Institute*
OSIRIS at Van Andel Institute will enable VAI bioinformaticians to work with MSU researchers to better understand Parkinsons disease and cancer, and will allow access to VAI researchers with MSU appointments to access the computational resources at MSU ICER. The OSiRIS site at Van Andel is deployed and managed similar to other OSiRIS sites. The 3 nodes there are part of the multi-institutional OSiRIS cluster and OSD are partitioned into a separate Ceph Crush tree to be used in rules defining cache tier pools. More architectural details of the Ceph cache tier deployment at Van Andel are covered in section 3.1.1.

*5.5. Engagement Roadmap*
For reference, this was our initial road-map for science domain engagement. In the course of the project there has been some deviation from this depending on the readiness of planned science domains and inquiries from new science domains. We have also had the capacity to enroll new science domains much earlier than planned.

- End Year 1: High-energy Physics, High-resolution Ocean Modeling
- End Year 2: Biosocial Methods and Population Studies, Aquatic Bio-Geochemistry
- End Year 3: Neurodegenerative Disease Studies
- End Year 4: Statistical Genetics, Genomics and Bioinformatics
- Year 5: Remaining participants, New Science Domains

## 6. Deploying Your Own OSiRIS

Up to this point we have discussed the details of our prototype deployment of an OSiRIS infrastructure. In this section we will summarize the process of deploying an OSiRIS infrastructure in a new location.

*6.0.1. Hardware Considerations* One of the most important considerations will be your storage node configuration. The following criteria should inform your hardware purchases:

- Aim for one CPU core, including SMT cores, per OSD process on the system. Slightly less may be acceptable if you are constrained by other limitations such as latency between sites. This is not simply a performance consideration. Insufficient CPU can result in your cluster catastrophically failing when under heavy load as might occur during recovery from node failures or heavy client usage.
- Generally the best cost/performance ratio is by using hybrid OSD that combine a single HDD with NVMe for Bluestore DB. The amount of NVMe space is recommended to be 4 percent of each OSD (multiplied by number of OSD on the system). Because of how the underlying storage metadata DB levels are scaled you may not see benefits from less NVMe DB space[40].
- Ceph OSD default to using 3-5GB of memory but can be tuned higher if you have it available[40]
- You may wish to deploy some higher-performance nodes with NVMe-only storage. Our experience and the general recommendation is that an NVMe disk should host multiple OSD to utilize the full performance capacity[41].

The sizing and configuration of your storage nodes will be heavily dictated by the above constraints. Very dense nodes, such as the 60-disk JBOD used by OSiRIS, require per-node very high CPU core counts, memory, and NVMe storage. Keep in mind your cluster must also be able to tolerate the loss of one or more nodes. Downtime for a very large node equates to a very large amount of data now needing to find new locations for replicas. Free space must be left to accommodate the loss of an entire node, and ideally multiple nodes. Otherwise it is possible to find the cluster unable to function due to loss of a single machine.

In general our experience would dictate that it is preferable to deploy a larger number of less dense nodes though this must be balanced with the per node cost. You may want to ask vendors to quote several potential configurations meeting the above requirements for various counts of disks.

Further recommendations for storage and for other Ceph components are available from the Ceph documentation[42]. We also recommend joining and searching the ceph-users email list for discussions of hardware sizing[43].

Components such as virtualization nodes, Globus, RGW endpoints, NFS servers, perfSonar, etc will depend on your expected use. You will want to review documentation available for these pieces to determine appropriate hardware or VM sizing.

*6.0.2. Network Planning* Network latency and bandwidth very much determines your maximum Ceph performance. You can avoid replicating pools over high latency or low bandwidth links by using CRUSH to place replicas as appropriate, or delay replicating data by using a cache tier. Regardless, the links between the OSD containing data replicas determine your maximum performance as much or more than any hardware configuration.

In a multi site deployment the most ideal configuration is to have equivalent same-speed same-latency links between the member sites. For other situations you will have to look into throttling Ceph recovery operations as covered in 3.1.3. You may still encounter bottlenecks under heavy client load though we have not yet seen enough simultaneous usage to encounter this issue. We do not have a recommendation at this point for settings to limit Ceph performance at the client level, and any such limit would likely be detrimental to the user experience.

Given a cluster of hosts with aggregate bandwidth exceeding the available WAN bandwidth, Ceph can fully utilize inter-site links in normal operations. As such it is recommended you work with your campus networking team to establish direct, dedicated links for the storage components of your platform when replicating pools across multiple sites (see also section 3.3).

*6.0.3. Bootstrapping a New Deployment* To begin a new OSiRIS-like deployment the process is roughly as follows:

- Some kind of host or virtual machine is needed to deploy Foreman and Puppet. In our case we started with a machine intended to host a variety of virtual machines and used Libvirt tools packaged with Linux-based OS to create a virtual machine. Once the machine was configured with an OS we initialized Foreman and Puppet using the Foreman-installer tool[44]
- Once puppet is available it can be linked to git and basic configuration modules can be included in the puppet environment[32]. Begin by applying this configuration to the Puppet/Foreman master host and (if relevant) the hardware hosting the VM. Consider deploying some test boxes to develop a basic configuration and process for deploying from Foreman. Various puppet modules are available from the Puppet Forge[45].
- After this initial deployment, additional sites do not require the full Foreman/Puppet setup. Instead a host can be deployed at these sites and configured as a Foreman smart proxy (see section 4.1). The initial host will have to be manually installed with an OS but can then

be setup with Puppet configuration pulled from the first site. Use Foreman modules from the Puppet Forge to configure the provisioning proxies.

- Once the core configuration and provisioning tools are in place you can begin to build Ceph components. Begin with a basic configuration with 3 or 5 Ceph monitors and managers to which you can add OSD, MDS for CephFS, Radosgw, etc. Use Puppet modules for Ceph available from OSiRIS[46] or from the upstream module that we forked from.
- Once a Ceph cluster is functional you can begin to add additional components and monitoring. These might include Globus endpoints, NFS servers, HAproxy or standalone S3 endpoints, perfSonar network testpoints, and metric collection (ceph-mgr or collectd plugins). Options for these components are discussed elsewhere in this paper.

*6.0.4. Authentication Federations* Before beginning to deploy AAA components you may want to start the process of joining your local federation if your desire is to provide resources outside your institution. This is a process that will generally occur at the institutional level[47] though it is possible at other levels of organization. Procedures may vary for other federations such as eduGain. When deploying OSiRIS we worked with our institutional identity management teams to register our Shibboleth Service Provider (COmanage host) with InCommon and part of this process was also registering with eduGain.

If possible you may also want to look into avoiding Shibboleth and using CILogon to authenticate external users to COmanage or other service deployments. This is necessarily vague advice because we have only interacted with CILogon as it relates to Globus user authentication but in general Shibboleth is less preferable to more modern solutions based on OAuth or OIDC. CILogon can provide user identity via these protocols without the overhead of configuring your own Service Provider and registering it with a federation. Signing in with CILogon still requires InCommon federation membership as it is actually a bridge from SAML-based federations to various means of issuing x.509 certificates or OIDC tokens.

*6.0.5. Deploying VO Management* You will need to deploy a replicated LDAP server as discussed in section 3.5.4. You are then ready to begin deploying COmanage and Grouper and tie these into LDAP. In the case of Grouper we have a Puppet module available[48] but you may find it more convenient to deploy containers provided by the Grouper project[49]. You can reference our Puppet module for example configuration but our LDAP configuration is not particularly different from the Grouper documentation examples. COmanage requires a manual installation following instructions available from their website[23]. Once you have a working installation of COmanage you are ready to install the Ceph Provisioner plugin as described in section 3.5.2.

## 7. Next Steps

In the final years of the project we consider the following enhancements to be most important:

- Continuing to expand on our toolkit of client interfaces, an working with science domains to help them most efficiently leverage object storage. Many scientists are used to working with on-campus Posix storage. We can certainly offer some of those capabilities but for wider collaboration and computing in-place on batch cluster systems they would be better served to use S3 to work with the data.
- Implementing in production settings our work in software-defined networking (SDN) orchestration of both science-user and OSiRIS infrastructure network connectivity.
- Providing easily re-usable templates for deploying services similar to OSiRIS in the form of Puppet 'profile' classes. Such profiles are used internally as wrappers to include necessary puppet modules and config to those modules which dictates our standard setups.

- Developing automated data life-cycle meta-data creation for users of OSiRIS based on the data we are now gathering from the research community.
- Transitioning the project into a campus operated service without dedicated external funding. All of our member institutions have committed to continuing OSiRIS if it is seen to be a useful service.

## 8. Conclusions

The OSiRIS project goal is to enable scientists to collaborate on data easily and without building their own infrastructure. Scientists should be able to use our infrastructure by leveraging their existing institutional identities for authentication and self-management of resources. We aim not only to provide a scalable shared storage infrastructure, but to enable the most efficient use of that infrastructure with active network management via our NMAL layer. Users of OSiRIS should be able to get science done with their data instead of becoming bogged down in the details of data management and access.

## Acknowledgments

We gratefully acknowledge the National Science Foundation (grant 1541335) which supported the work in this paper.

## References

[1] Weil S A, Brandt S A, Miller E L, Long D D E and Maltzahn C 2006 Ceph: a scalable, high-performance distributed file system (USENIX Association) pp 307–320 URL http://www.ssrc.ucsc.edu/papers/weil-osdi06.pdf
[2] Weil S A, Brandt S A, Miller E L, Long D D E and Maltzahn C 2006 Crush: controlled, scalable, decentralized placement of replicated data 122 (ACM New York) URL http://www.ssrc.ucsc.edu/papers/weil-sc06.pdf
[3] 2019 Cache tiering URL http://docs.ceph.com/docs/master/rados/operations/cache-tiering
[4] 2016 Osiris at supercomputing 2016 URL http://www.osris.org/article/2016/11/18/osiris-at-supercomputing-2016
[5] 2019 Setting primary osd for read optimization URL http://www.osris.org/article/2019/03/01/ceph-osd-site-affinity
[6] OSiRIS 2019 Ceph URL http://www.osris.org/components/ceph
[7] 2018 Osiris at supercomputing 2018 URL http://www.osris.org/article/2018/11/19/osiris-at-supercomputing-2018
[8] 2015 Plugin:ceph URL https://collectd.org/wiki/index.php/Plugin:Ceph
[9] 2017 The influxdb ceph-mgr plugin URL http://www.osris.org/article/2017/12/07/the-influxdb-ceph-mgr-plugin
[10] 2019 Ceph manager daemon URL http://docs.ceph.com/docs/master/mgr
[11] 2013 Nfsv4 in a multi-realm environment URL http://www.citi.umich.edu/projects/nfsv4/crossrealm/libnfsidmap_config.html
[12] 2018 ldaputil: Misc ldap utilities used in osiris URL https://github.com/MI-OSiRIS/ldaputil
[13] 2019 Amazon s3 rest api introduction URL https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html
[14] 2019 Haproxy unix socket interface URL https://cbonte.github.io/haproxy-dconv/1.7/management.html#9.3
[15] Mkrtchyan T 2017 dcache + ceph URL https://www.dcache.org/manuals/workshop-2017-05-29-Umea/000-Final/tigran-dcache+ceph.pdf
[16] Data sharing with globus URL https://www.globus.org/data-sharing
[17] 2018 Globus connect server v4 installation guide URL https://docs.globus.org/globus-connect-server-installation-guide
[18] 2018 voperson: Attribute management within a virtual organization URL https://voperson.org
[19] 2018 Globus cilogon subject dn ldap query URL https://groups.google.com/a/globus.org/forum/#!topic/admin-discuss/8D54FzJzS-o
[20] 2017 Incommon URL https://www.incommon.org/
[21] 2019 What is edugain URL https://edugain.org/about-edugain/what-is-edugain

[22] Cantor S e a 2005 Shibboleth architecture: Protocols and profiles URL https://wiki.shibboleth.net/confluence/download/attachments/2162702/internet2-mace-shibboleth-arch-protocols-200509.pdf
[23] Olshansky S 2017 Comanage URL https://spaces.internet2.edu/display/COmanage
[24] 2019 Grouper URL https://www.internet2.edu/products-services/trust-identity/grouper
[25] 2018 Ceph provisioner plugin for comanage URL http://www.osris.org/components/cephprovisioner.html
[26] 2018 Introduction to sssd URL https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system-level_authentication_guide/sssd
[27] 389 directory server URL https://directory.fedoraproject.org/index.html
[28] 2017 puppet-ds389: A module to configure 389 ds and multimaster replication URL https://github.com/MI-OSiRIS/puppet-ds389
[29] Configuring replication from the command line URL https://access.redhat.com/documentation/en-us/red_hat_directory_server/10/html/administration_guide/managing_replication-configuring-replication-cmd
[30] 2019 Foreman plugins URL https://www.theforeman.org/plugins
[31] 2019 Iso and usb boot disk support for foreman URL https://github.com/theforeman/foreman_bootdisk
[32] Larizza G 2014 Building a functional puppet workflow part 3: Dynamic environments with r10k URL http://garylarizza.com/blog/2014/02/18/puppet-workflow-part-3/
[33] 2019 Nfs ganesha: Nfs ganesha container designed to serve cephfs or ceph rgw shares URL https://hub.docker.com/r/miosiris/nfs-ganesha-ceph
[34] 2019 Miosiris docker hub community URL https://hub.docker.com/r/miosiris
[35] 2018 Slate: Service layer at the edge and mobility of capability URL http://slateci.io
[36] 2016 Dynafed - the dynamic federation project URL http://lcgdm.web.cern.ch/dynafed-dynamic-federation-project
[37] Adams J, Canning B, Dewhurst A, Johnson I, Packer A and Vasilakakos G 2016 Building a large scale object store for the ral tier 1 URL http://indico.cern.ch/event/505613/contributions/2230932/attachments/1346825/2039423/Oral-v2-556.pdf
[38] 2018 Osiris research highlight: Physical oceanography simulations URL http://www.osris.org/domains/ocean.html
[39] 2018 Osiris research highlight: Effect of the placental epigenome on stunting in a longitudinal african cohort URL http://www.osris.org/domains/genome-stunting.html
[40] Bluestore configuration reference URL http://docs.ceph.com/docs/master/rados/configuration/bluestore-config-ref
[41] Tuning for all flash deployments URL http://tracker.ceph.com/projects/ceph/wiki/Tuning_for_All_Flash_Deployments#NVMe-SSD-partitioning
[42] Ceph hardware recommendations URL http://docs.ceph.com/docs/master/start/hardware-recommendations
[43] Ceph users email list URL http://lists.ceph.com/listinfo.cgi/ceph-users-ceph.com
[44] 2019 Foreman installer URL https://theforeman.org/manuals/1.21/index.html#3.InstallingForeman
[45] Puppet forge URL https://forge.puppet.com
[46] 2018 Puppet ceph module URL https://github.com/MI-OSiRIS/puppet-ceph
[47] Joining incommon URL https://www.incommon.org/join.html
[48] 2019 Puppet grouper module URL https://github.com/MI-OSiRIS/puppet-grouper
[49] Tier grouper docker container URL https://hub.docker.com/r/tier/grouper