Wellcome Trust Genome Campus
Hinxton Cambridge CB10 1SA
T 01223 834244
F 01223 494919
www.sanger.ac.uk

Introduction

Current scientific software and HPC applications rely heavily upon performant, shared POSIX compliant filesystems. The combination of ever larger data-sets, the personal information that they may contain and the evolving privacy laws mean that it is more challenging than ever before to meet the legislative requirements and the high performance access to data at scale, which makes scientific research possible.

The bioinformatic pipelines at Sanger are no exception. Addressing large-scale scientific computation challenges with appropriate data access and cross-group restrictions requires solutions that can be applied to both current and developing IT and scientific instrument technologies. There is a clear requirement for a performant multi-tenant high performance clustered file system with a relatively low barrier to entry and using existing filesystem features, wherever possible.

We will explain how to provide a project with access to a Lustre filesystem, where that access is restricted to a subdirectory, and users and groups are mapped to a single identity. We will show how this can be done for multiple projects.

Introducing OpenStack (for Lustre users)

OpenStack is an open source system which allows users to create and manage virtual machines and their associated infrastructure using both APIs and a web interface. Supported deployments are available from Red Hat, Mirantis, Canonical, StackHPC and others.

Software-defined routers route packets between various networks. There are a number of different types of network in OpenStack:

- Self service networks are created dynamically and can have any IP address range the user requires. Machines provisioned on a self service network will typically have an IP address automatically assigned to them by DHCP. These are implemented as either a VLAN or via encapsulation e.g. GENEVE or VXLAN. Self service networks are owned by a project in openstack.
- The public network is special as it is connected to the outside world - either as a true public network or a routable network inside an organisation if a cloud is available for private access only. This is implemented as a VLAN as it is used to interface with standard networks. Public networks are owned by the OpenStack administrator.
- Provider networks are used by the cloud provider to provide services to projects and are typically implemented as VLAN networks. Access to each provider network can be controlled by roles allocated to a project. Public networks are owned by the OpenStack administrator.

Sanger's networking for OpenStack is provided by Arista switches in a leaf-and-spine arrangement, with bonding/port-channel used to provide high-availability connections. Compute nodes (hypervisors) are connected at 2x25GbE; Ceph, controller and networker nodes have 2x100GbE. The Sanger institute has two main OpenStack deployments:

1. A general purpose OpenStack which is based on a Kolla "Train" release with support from StackHPC. This provides approximately 10,000 threads, and 115TB of RAM, with access to a 5PB usable Ceph cluster providing block (Cinder) and S3-compatible (radosgw) access.

2. A dedicated production sequencing (currently used for Covid and UK Biobank) OpenStack instance based on Red Hat OpenStack Platform 13, which is based on the upstream “Queens” release. This has approximately 3,500 threads, and 36TB of RAM, with access to a 70TB usable Ceph cluster providing block (Cinder).

Introducing Lustre (for OpenStack users)

Lustre is an open-source high performance POSIX filesystem which has been in use and development for over eighteen years and in use at Sanger for over thirteen years. It combines multiple servers for both metadata and block services to provide a single coherent filesystem. High performance is achieved by reading and writing from a large number of servers and disk groups in parallel. Commercial deployments can be bought from Data Direct Networks, HPE, Dell and others.

There are a pair of features (subdirectory mounts and UID/GID mapping) introduced in Lustre 2.9 that can be used to provide a multi-tenant POSIX filesystem.

Lustre is supported on various network technologies and a protocol called LNet is used on top of the physical network. A Lustre address looks like `192.168.11.227@tcp0` and is called a NID, or network identifier. `tcp0` is the Lustre network label[1] while the string before the @ is used as an IP address and it is assumed that any two Lustre nodes on the same LNet label can route IP packets to each other. While it would be possible for two nodes on different Lustre networks to have the same IP address it would be confusing and is not best practice. Lustre cannot be used through a NAT gateway.

A nodemap[2] is a mapping of a collection of IP addresses ranges on a Lustre network which have a shared set of access limitations. For example all access might be mapped to user id 32769 and mount requests would only allow access to the project25 subdirectory.

There are three separate service types for Lustre, each of which supports failing over to a number of different servers to support high availability.

- The MGT (Management Target) is a configuration store and is used to store the configuration of the filesystem including the network addresses of the other server components. Each filesystem has a single MGT. A single server (MGS) can be used for multiple filesystems; however we have decided not to do this as it makes upgrades complex when you have multiple filesystems on a single MGS.
- The MDT (MetaData Target) stores the attributes of files in the filesystem. A Lustre filesystem can have multiple MDTs, these can be used to increase the metadata performance of a filesystem. A single server (MDS) can host multiple MDTs. A Lustre filesystem typically has a small number of MDTs, 1-4 hosted on two MDSs. Recent changes to the MTD allows the first segment of a file to be stored on the MDT.
- The OST (Object Storage Target) stores the blocks of the filesystem. A single server (OSS) can provide multiple OSTs. A typical filesystem at Sanger might have 12 OSTs, hosted on 6 OSSs.

The Sanger Institute has been using Lustre for over 15 years and the filesystem capacity currently in use exceeds 25PB. It is entirely Ethernet-connected, with each Lustre component server typically having multiple 100GbE connectivity in the most recent installations.

[1] http://doc.lustre.org/lustre_manual.xhtml#idp694976 Lustre operations manual section 2.3 Lustre Networks.
[2] http://doc.lustre.org/lustre_manual.xhtml#lustrenodemap Lustre operations manual chapter 24

Nodemaps

DDN/Whamcloud currently supports 32 nodemaps per filesystem. One nodemap is used per OpenStack project and this limits the number of projects that can access a Lustre filesystem. The Sanger have just under 200 projects while CERN have just over 4000 projects[3]. A nodemap has the following attributes which we specify:

fileset	The directory that clients that have access to this nodemap can access.
deny_unknown	If set to true then access is denied to files who's user is not mapped.
uids	A list of UID pairs one in of the pairs in the canonical space and one in the mapped space eg. 1000:20000 this would map UID]1000 in the secure Lustre space to uid 20000 in the canonical space.
squash_uid	The default UID that users will be squashed to if unmapped, unless the deny_unknown flag is set, in which case access will still be denied.
gids	A list of GID pairs one in of the pairs in the canonical space and one in the mapped space eg. 2000:40000 this would map gid 2000 in the secure Lustre space to GID 40000 in the canonical space.
squash_gid	The default GID that users will be squashed to if unmapped, unless the deny_unknown flag is set, in which case access will still be denied.
admin	The "admin" property defines whether root is squashed on the nodemap members.
trusted	The "trusted" property permits members of a policy group to see the file system's canonical identifiers, e.g. this allows the nodemap members to see the real UIDs.
ranges	Each nodemap has a list of ranges which allow access to the nodemap for example: 10.177.[64-65].[0-255]@tcp7
audit_mode	When set to 0, events are not logged into the Changelogs, no matter if Changelogs are activated or not.
map_mode	Valid values are both,uid_only,gid_only

LNet space

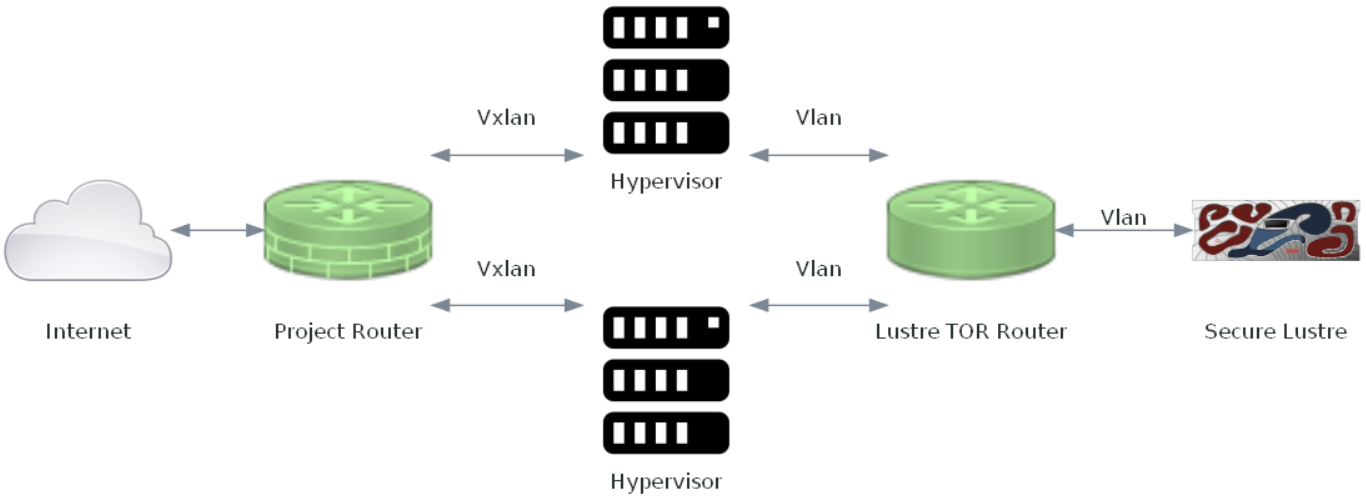
DDN/Whamcloud support 17 Lustre network labels, which are traditionally named tcp0 -> tcp16. Labels were originally implemented for infiniband where high performance clusters had multiple networks. The same idea could be used in Ethernet networks with each label being used to denote a separate ipv4 address space.

Sanger uses tcp0 for the trusted HPC connection as this is the label we have always used. This means that a HPC client can access a secure Lustre system with the same configuration as it uses to access a legacy Lustre system. Originally Sanger was expecting to have a separate LNet space per project however we discovered that Lustre only supports 17 namespaces and so another solution had to be created. After some thought it became clear that each project did not need a separate LNet space but

[3] https://superuser.openstack.org/articles/cern-openstack-update/

only needed a distinct LNet/IP network range. Sanger decided that each division eg, Human Genetics, Cancer Aging Somatic Mutations etc should have its own LNet space, the reason being that it would be easy to demonstrate if there was an issue with the ip address matching in Lustre then the data exposed would always at least be in the same division. At this time we probably would not make the same choice again as I do not believe that the potential added checks are worth the complexity that is required. At the very least two LNet spaces are required, one for HPC access and the other for OpenStack. It is possible to migrate between our current configuration and a new configuration by creating new networks in a single LNet and granting access to these networks to the OpenStack projects.

Network topologies



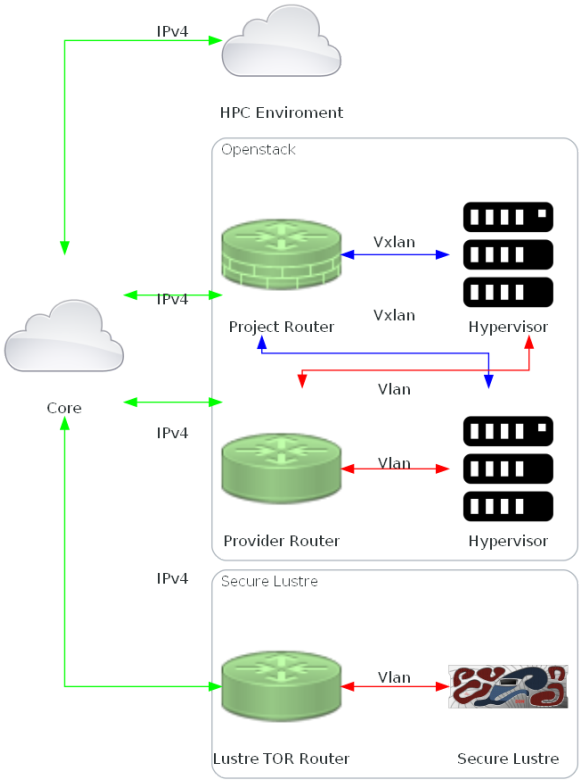
We made the decision that any OpenStack Lustre client would have two virtual Ethernet interfaces, a normal self service network for data and a provider network which is solely used for lustre traffic. While it would be possible to use a single provider network, the SDN features of OpenStack would be lost and the access control lists (ACL) needed to be implemented would be overly complex. It is the responsibility of the layer three switches in the network system to ensure that packets from a secure Lustre client are only allowed to be delivered to the correct IP network. There is a general principle that ACLs should be applied as close to the packet generation as possible. This means that the ACLs protecting the storage need to be applied on the Lustre top of rack network router, while the provider network ACL should be applied to the SVI (Switched Virtual Interface - the layer three interface on a Vlan implemented by a layer three switch). Sanger uses Arista 7060 which are 100Gbit/second capable switches (each 100Gbit port can be broken out into four ports of 25Gb/second).

The simplest logical network topology is where both the storage and the Lustre clients are connected to a single router. In this topology the Lustre “top of rack” router is connected to the storage and to the hypervisors and implements all the ACL’s. This topology is not feasible due to the limited number of ports available on commodity switches.

A more realistic logical topology would be where the Lustre top of rack is connected to the campus core and each of the provider networks is also connected to a router which is then connected to the campus core. This topology is closer to what we actually use.

ACL’s are implemented on the Lustre top of rack router to protect the Lustre appliance’s management interfaces and to ensure that only packets sourced from the correct Lustre networks are allowed to arrive at the sub interfaces on the secure Lustre appliance.

ACL's are implemented on the provider router to ensure that the provider network can only send packets to the network associated with the Lustre label on the secure Lustre appliance.

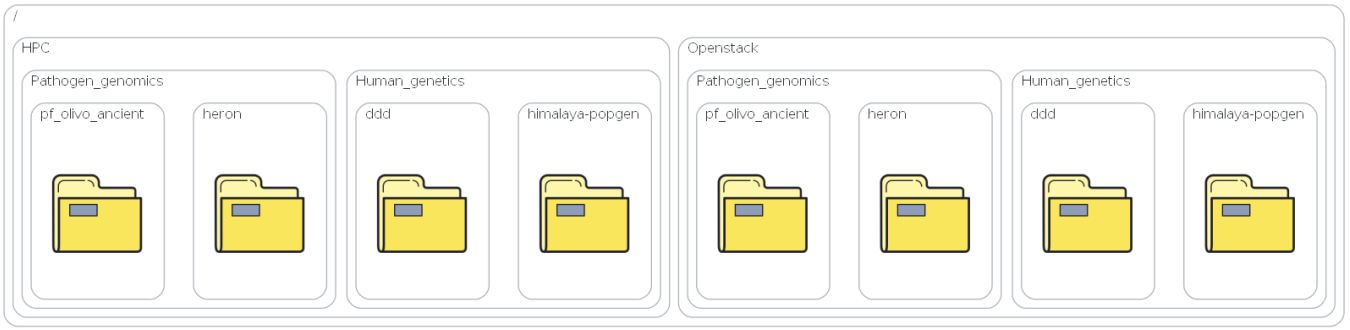


A more detailed description of our production networking can be found in Appendix A.

Directory layout

The first decision to be made is whether the traditional HPC access should be allowed to access the whole file system or not. If full access is not granted to HPC clients then a management/admin project should be created to allow the Systems Administrators access to the full system with the ability to create multiple machines if there are operations that require significant IO capability.

The Sanger Institute has projects which have a common organisational unit, for example Human Genetics or Pathogen Genomics. By creating directories for the organisational unit it is possible for the administrators for each unit to have access to all their unit's data with on virtual machines that they control. Whether root access to the sub directories is granted is a matter of data policy.



Any of the boxes is a reasonable directory to share to an OpenStack project. If you don't restrict access to the root directory to the HPC systems then you cannot guarantee that data shared with a project in OpenStack can not be altered by an Administrator of the HPC system. At Sanger the HPC and OpenStack team are synonymous.

Identity mapping

The problem that we are trying to address is trusting the client. Users working from HPC do not have “root” access so POSIX user/group permissions are adequate, whereas those working from inside OpenStack do - and can therefore switch to any user or group at will. Currently any member of an OpenStack project has access to all data accessible from the OpenStack project - instances, images and volumes - and the membership of the OpenStack project MUST therefore align with the data access policy. Secure Lustre relies on this alignment and has no provision for “subtenants” where a user in a project must not have access to data belonging to another user in the same project.

Sanger specifies that the mapped user id used must be a “pipeline” account that is not tied to an individual.

To help explain the various nodemap settings we created a secure Lustre area and placed a number of files in them and created a local user jb23 whose local gid is 1000. The following settings are constant:

uid map	client_id: 1000 (ubuntu) is mapped to fs_id 15230 (isgbot)
gid map	client_id: 1000 (ubuntu) is mapped to fs_id 1533 (isg)
squash uid	15230
squash gid	1533

These are the settings we often use.The most relevant point is that / of the share is owned by the mapped group and therefore an OpenStack user can remove all the data in the share as they have write permission to the directory. As the local user jb23 has also gid 1000 then that user has access to the files via group permissions.

Openstack	HPC
ubuntu@jb23manops:/lustre/scratch123$ ls -l total 16 -rw-r--r-- 1 ubuntu ubuntu 2 Apr 23 14:49 1 -rw-r--r-- 1 ubuntu ubuntu 2 Apr 23 14:49 2 -rw-r--r-- 1 jb23 ubuntu 2 Apr 23 14:49 3 -rw-r--r-- 1 jb23 1533 2 Apr 23 14:49 4 ubuntu@jb23manops:/lustre/scratch123$ rm -f 3 ubuntu@jb23manops:/lustre/scratch123$ ls -l total 12 -rw-r--r-- 1 ubuntu ubuntu 2 Apr 23 14:49 1 -rw-r--r-- 1 ubuntu ubuntu 2 Apr 23 14:49 2 -rw-r--r-- 1 jb23 1533 2 Apr 23 14:49 4 ubuntu@jb23manops:/lustre/scratch123$	root@farm5-head1:/lustre/scratch123/admin/tea m94/openstack/k8_test# ls -l total 16 -rw-r--r-- 1 isgbot ssg-isg 2 Apr 23 15:49 1 -rw-r--r-- 1 isgbot ssg-isg 2 Apr 23 15:49 2 -rw-r--r-- 1 jb23 ssg-isg 2 Apr 23 15:49 3 -rw-r--r-- 1 jb23 jb23test 2 Apr 23 15:49 4 root@farm5-head1:/lustre/scratch123/admin/tea m94/OpenStack/k8_test# ls -l total 12 -rw-r--r-- 1 isgbot ssg-isg 2 Apr 23 15:49 1 -rw-r--r-- 1 isgbot ssg-isg 2 Apr 23 15:49 2 -rw-r--r-- 1 jb23 jb23test 2 Apr 23 15:49 4 root@farm5-head1:/lustre/scratch123/admin/tea m94/openstack/k8_test#

What can be seen is that all access to the file system is as uid 1000 and gid 1000 however full users and group information is being sent to the client and it can interpret it for example as we have a local entry in /etc/passwd for jb23 the secure lustre server can display the correct username.

If you want to protect an area from a project do not set the ownership of the root of the share to mapped group, this way you can have a directory containing hard links to reference files and these should not be changeable by the OpenStack users.

Openstack	HPC
ubuntu@jb23manops:/lustre/scratch123$ ls -lR .: total 8 drwxr-sr-x 2 jb23 1533 4096 Apr 26 15:53 reference drwxrwsrwx 2 jb23 ubuntu 4096 Apr 23 14:48 write ./reference: total 33554444 -rw-r--r-- 4 jb23 1533 34359738368 Oct 14 2020 real_reference lrwxrwxrwx 1 jb23 1533 21 Apr 26 15:48 reference -> ../../../test/bigfile ./write: total 16 -rw-r--r-- 1 ubuntu ubuntu 2 Apr 23 14:49 1 -rw-r--r-- 1 ubuntu ubuntu 2 Apr 23 14:49 2 -rw-r--r-- 1 jb23 ubuntu 2 Apr 23 14:49 3 -rw-r--r-- 1 jb23 1533 2 Apr 23 14:49 4 ubuntu@jb23manops:/lustre/scratch123$ rm reference/reference rm: cannot remove 'reference/reference': Permission denied ubuntu@jb23manops:/lustre/scratch123$ rm -f write/3	root@server:# chown jb23:jb23test . root@server:# mkdir reference root@server:# cd reference/ root@server:/reference# ln -s ../../../test/bigfile ./reference root@server:/reference# ln ../../../test/bigfile ./real_reference root@server:# ls -lR .: total 8 drwxr-sr-x 2 root jb23test 4096 Apr 26 16:53 reference drwxrwsrwx 2 hc7 ssg-isg 4096 Apr 23 15:48 write ./reference: total 33554444 -rw-r--r-- 4 jb23 root 34359738368 Oct 14 2020 real_reference lrwxrwxrwx 1 root jb23test 21 Apr 26 16:48 reference -> ../../../test/bigfile ./write: total 16 -rw-r--r-- 1 isgbot ssg-isg 2 Apr 23 15:49 1 -rw-r--r-- 1 isgbot ssg-isg 2 Apr 23 15:49 2 -rw-r--r-- 1 jb23 ssg-isg 2 Apr 23 15:49 3 -rw-r--r-- 1 jb23 jb23test 2 Apr 23 15:49 4

Now we have established that the hard links and group ownership of directories work as expected. We can continue with how the flags on the share change the data that is visible.

admin_nodemap:0, deny_unknown:0, trusted_nodemap:0	
Openstack	HPC
ubuntu@jb23manops:/lustre/scratch123$ ls -la total 24 drwxrwsrwx 2 jb23 ubuntu 4096 Apr 23 14:48 . drwxr-xr-x 4 root root 4096 Oct 15 2020 .. -rw-r--r-- 1 ubuntu ubuntu 2 Apr 23 14:49 1 -rw-r--r-- 1 ubuntu ubuntu 2 Apr 23 14:49 2 -rw-r--r-- 1 jb23 ubuntu 2 Apr 23 14:49 3 -rw-r--r-- 1 jb23 1533 2 Apr 23 14:49 4	root@farm5-head1:/lustre/scratch123/admin/team94/OpenStack/k8_test# ls -la total 24 drwxrwsrwx 2 hc7 ssg-isg 4096 Apr 23 15:48 . drwxr-sr-x 3 root team94 4096 Apr 23 15:49 .. -rw-r--r-- 1 isgbot ssg-isg 2 Apr 23 15:49 1 -rw-r--r-- 1 isgbot ssg-isg 2 Apr 23 15:49 2 -rw-r--r-- 1 jb23 ssg-isg 2 Apr 23 15:49 3 -rw-r--r-- 1 jb23 jb23test 2 Apr 23 15:49 4
root@jb23manops:/lustre/scratch123# **touch root** root@jb23manops:/lustre/scratch123# **touch jb23** root@jb23manops:/lustre/scratch123# **chown jb23 jb23** root@jb23manops:/lustre/scratch123# **ls -l**	root@farm5-head1:/lustre/scratch123/admin/team94/OpenStack/k8_test# ls -la total 24 drwxrwsrwx 2 hc7 ssg-isg 4096 Apr 27 10:12 . drwxr-sr-x 3 root team94 4096 Apr 23 15:49 .. -rw-r--r-- 1 isgbot ssg-isg 2 Apr 23 15:49 1 -rw-r--r-- 1 isgbot ssg-isg 2 Apr 23 15:49 2

```
total 16
-rw-r--r-- 1 ubuntu ubuntu 2 Apr 23 14:49 1
-rw-r--r-- 1 ubuntu ubuntu 2 Apr 23 14:49 2
-rw-r--r-- 1 jb23   ubuntu 2 Apr 23 14:49 3
-rw-r--r-- 1 jb23     1533 2 Apr 23 14:49 4
-rw-r--r-- 1 ubuntu ubuntu 0 Apr 27 09:12
jb23
-rw-r--r-- 1 ubuntu ubuntu 0 Apr 27 09:12
root
```

```
-rw-r--r-- 1 jb23   ssg-isg     2 Apr 23 15:49 3
-rw-r--r-- 1 jb23   jb23test    2 Apr 23 15:49 4
-rw-r--r-- 1 isgbot ssg-isg     0 Apr 27 10:12
jb23
-rw-r--r-- 1 isgbot ssg-isg     0 Apr 27 10:12
root
```

Top of rack switch configuration

Access control lists on layer 3 switches are implemented in hardware using TCAM[4] by the switching ASIC in each switch. This is good from a performance point of view, however TCAM memory is a scarce resource and by carefully allocating IP addresses we can reduce the number of ACL entries that we need.

Given we have decided that each division should be allocated a LNet space we will need an ACL entry per LNet space if all the networks assigned to a division are adjacent. We allocate a /19 to each division initially which allows 16 networks (/23) each with 500 hosts. Each Lustre system requires an entry per division.The access control list is applied in the northbound direction on each software virtual interface of each provider network.

A more detailed discussion of our network switch configuration can be found in appendix B.

Lustre Image

ISG (Informatics Support Group) at the Sanger Institute supplies a number of virtual machine images to our user community, these images are created using continuous integration with packer and test-kitchen. While CI is configured for gitlab the code is available at CHANGEME and a copy of an image produced is available at CHANGEME. Of particular note is the lustre-config-setup service which runs before Lustre is loaded, it inspects the routing table and determines which Lustre it should have access to, creates hosts entries for the mgs and adds a commented fstab entries and configures the LNet module to use the correct interface. This script can be found at CHANGEME. Our configuration file for the Lustre mounting service (/etc/default/lustre-mount) is designed to be as generic as possible: it LNet pings each Lustre system using the hosts names configured earlier and if the ping command succeeds then it uncomments the line in /etc/fstab allowing the mount to happen.

[4] https://en.wikipedia.org/wiki/Content-addressable_memory#Ternary_CAMs

How to create a virtual machine

We have made the decision to use a separate storage network for Lustre traffic. The standard OpenStack server create command does not easily allow two networks with differing security policies to be created. We have a example set of bash scripts available [5] which demonstrate the method which:

1. Creates a port on the self service network with the specified security groups.
2. Creates a port on the provider network with port security disabled.
3. Creates a virtual machine configured to use a config-drive (we have found that using a config drive when machines have multiple interfaces to be more reliable) and the two ports created earlier.
4. Creates a floating IP address and binds it to the virtual machine.

OpenStack configuration

The OpenStack configuration has also been simplified, we now create the network and subnet, and then grant access to the network to a project.

```
openstack network create --provider-network-type vlan --provider-physical-network physnet1   --provider-segment 75
--no-share --internal --mtu 9000 --disable-port-security  lustre-npg01

openstack subnet create --dhcp --host-route destination=10.177.252.24/27,gateway=10.177.64.1 --network lustre-npg01
--allocation-pool start=10.177.64.10,end=10.177.65.240 --subnet-range 10.177.64.0/23 lustre-npg01

openstack network rbac create --target-project npg-esa --action access_as_shared --type network lustre-npg01
```

If we wanted to grant access to a single share to two projects then instead of granting access to both projects to a single network, create a second network and add the second network to the nodemap's ranges. This has the advantage that the provider network could not be used to communicate between the projects in an uncontrolled manner.

Lustre Servers

One of our lessons from our early experiments was that adding a project area piecemeal was prone to errors and not repeatable. This was especially true regarding adding LNet spaces. Our vendor installs Lustre systems with the maximum number of LNets configured. This does increase installation and testing time however a significant issue is sidestepped. We also now create the network and OpenStack configuration in multiples of 16 networks when each division first requires access or when a division requires another 16 project allocation.

To reduce the number of issues on the Lustre servers we implemented a simple bash script which allows us to have a common set of functions and network definitions and a set of default options for a nodemap definition and store this configuration in git. The script is available at CHANGEME and there should be a separate add_node_maps_filesystem.sh per filesystem. The networks used by each project are defined in networks.

One issue (LU-14657) we had to address was that if you configured all pieces of the nodemap policy at once the system did not apply the configuration consistently across all elements of the filesystem due to asynchronous updates. The script now creates the nodemap and sets the network

[5] https://github.com/HelenCousins/createserver

range, deny_unknown, trusted and admin policies and waits until these are set on the other MDSs before setting the fileset and identity mapping policies.

Performance

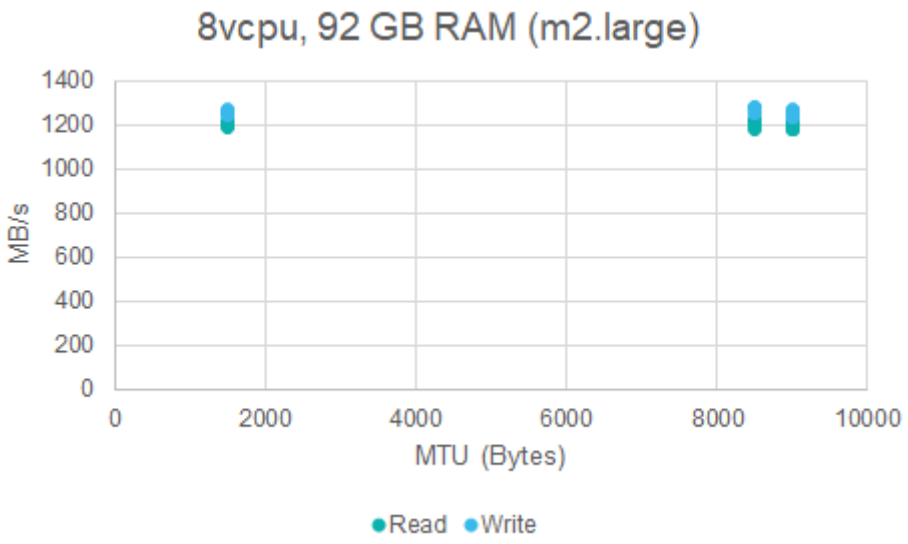
The Lustre file system used was scratch123 which is our latest research filesystem and is composed:

- 3 ES7990 each with a SS9012 enclosure
  - 2 OSS per ES7990 each with dual active/active 100GBits/second Ethernet.
  - 164 NL-SAS.
- 1 SFA200NV
  - 10 NVMe.
  - 8 active/active fibre channel (FC8).
- 2 Dell R640
  - Single Intel Xeon Platinum 8260, 192GB RAM.
  - Dual active/active 100GBits/second Ethernet.

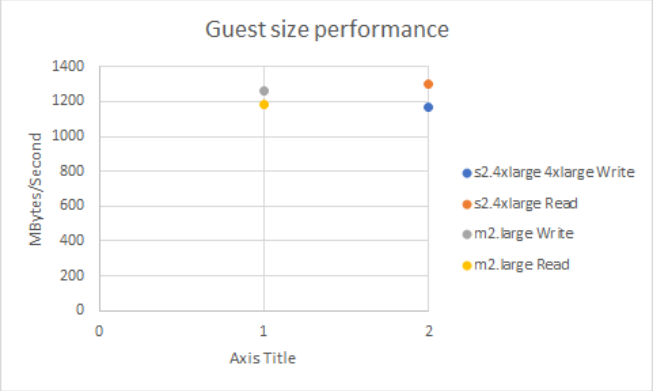
We did investigate using fio however we could not achieve consistent timings with the configuration file we had so the following shell script was used.

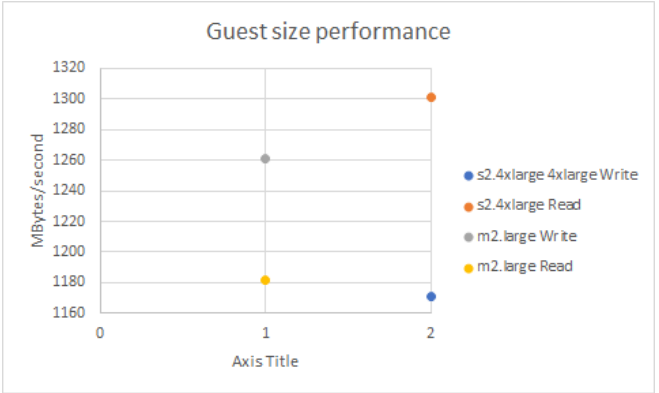
```
#!/bin/bash
rm -f deleteme
echo 3 > /proc/sys/vm/drop_caches
dd if=/dev/zero bs=1M count=4194304 status=progress conv=fdatasync of=deleteme
echo 3 > /proc/sys/vm/drop_caches
dd if=deleteme  bs=1M  status=progress  of=/dev/null
```

We expected MTU to change performance so we tested a virtual machine with a number of different MTU sizes. Performance was not significantly affected by MTU size.

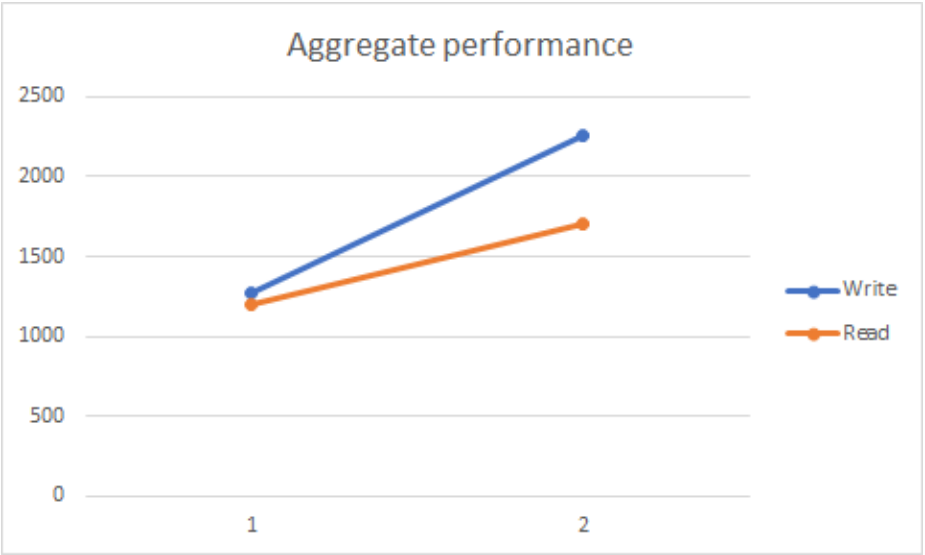


We wondered if instance sizes would affect performance, during our tests our small machine had no noisy neighbours on the hypervisor. Guest size does not appear to affect performance.

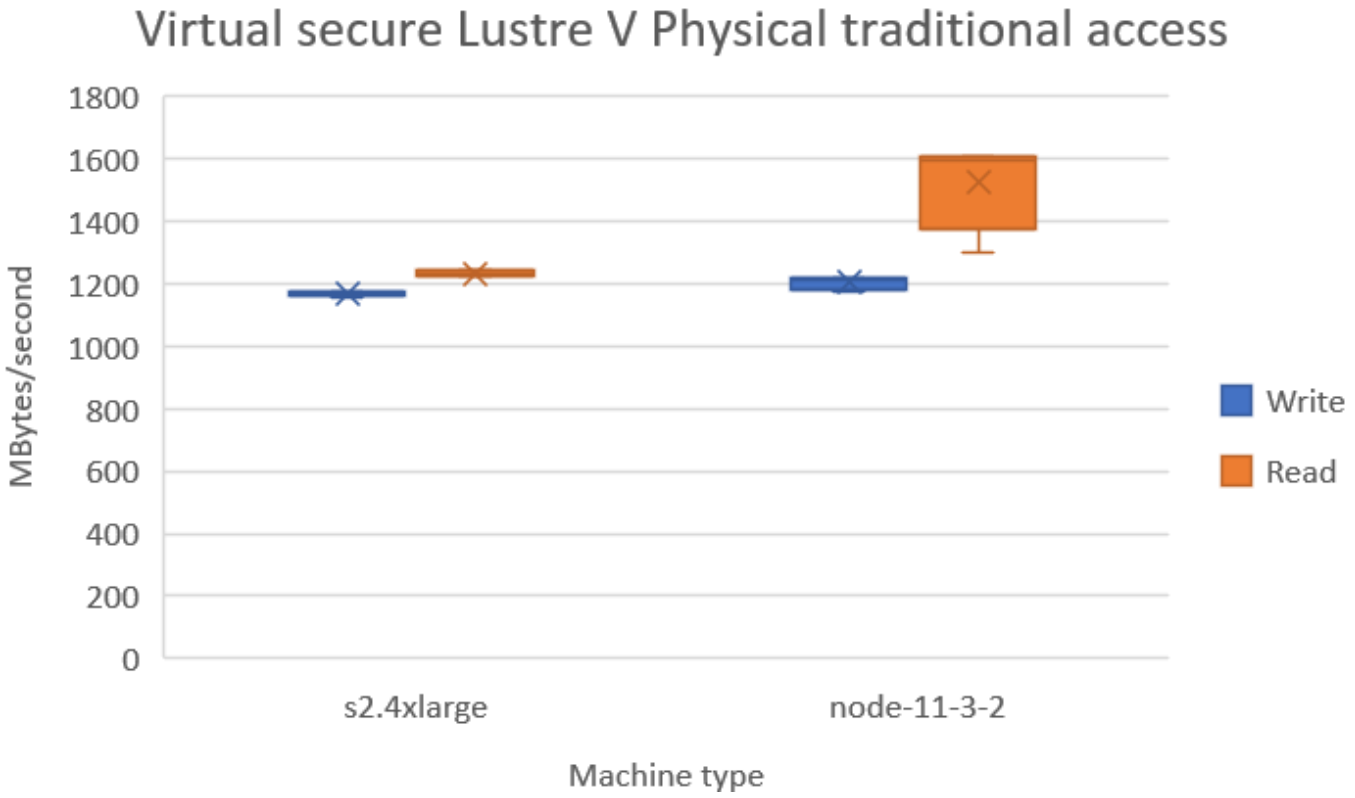




A test with two co located servers gave an increase in aggregate performance. However it was not clear if this was because our dd test was too simple and that performance was just gained via more streams,



We compared the performance between a full hypervisor guest and a hardware node and there is a significant difference in read performance. It is possible that the virtual read performance is not limited by the OVS performance while the write performance is.



Finally we configured a physical node to have the capacity to either connect conventionally and also additionally have the capability to connect via secure Lustre. The tests were repeated on this node to eliminate the performance variance of virtualisation.

From this we can see that there is no significant performance impact of accessing the filesystem via nodemaps and therefore there is no significant impact to the performance of using secure Lustre.

Future Work

We should investigate the idea of multiple mounts from a single Lustre system to provide read only access to reference genomes.

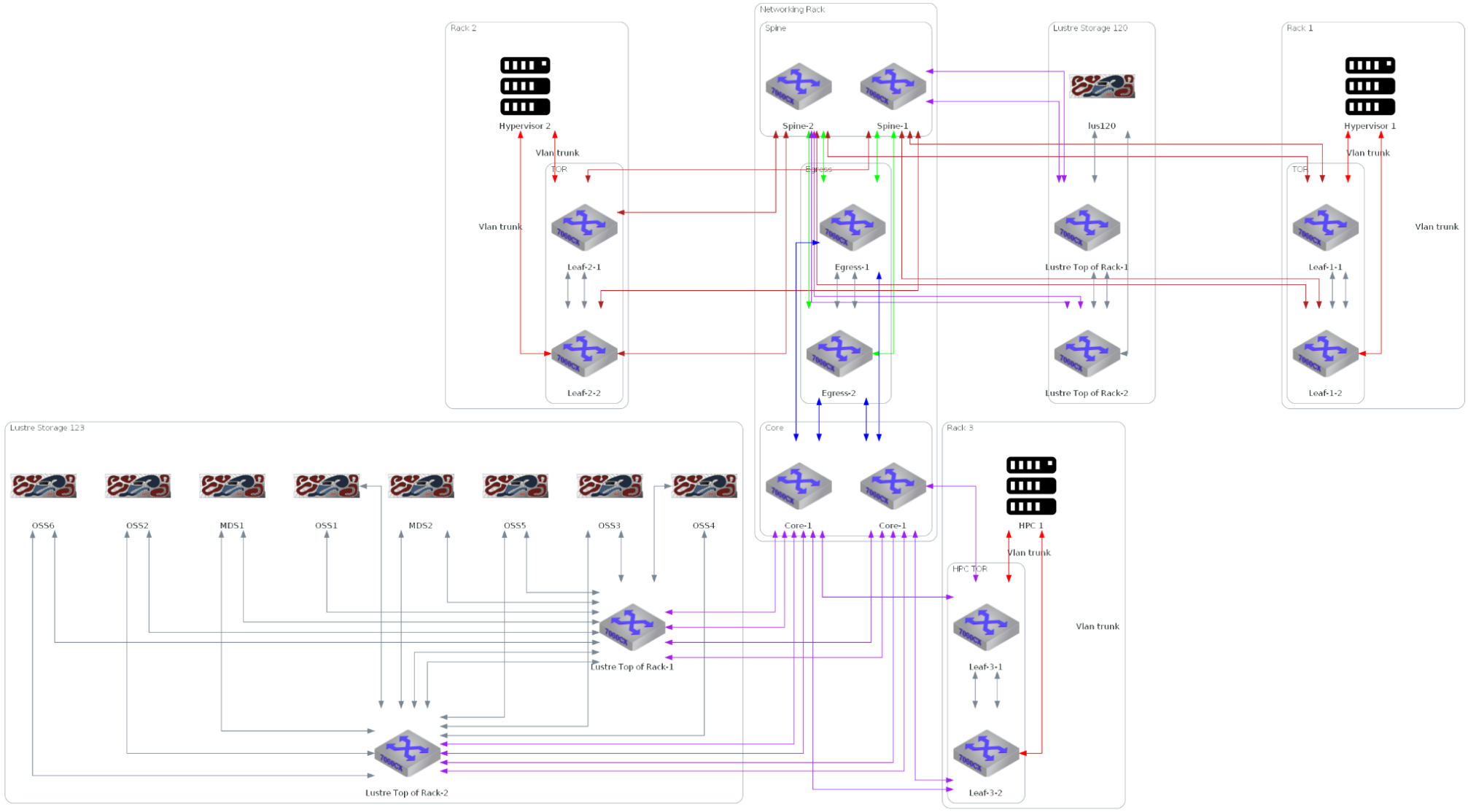
Conclusion

Secure Lustre using layer 3 routers provides a fast, secure access to a POSIX filesystem with no additional capital expense. Secure Lustre is used by our production sequencing group to process all our faculty sequencing.

Acknowledgements

Sanger Institute: James Beal, Jonathan Nicholson, Dave Holland, Helen Cousins, Peter Clapham, Matthew Venon, Tim Cutts, Pavos Antoniou, Christopher Harrison,Vivek Iyer.
DDN: Sébastien Buisson, Thomas Favre-Bulle, James Coomer, Richard Mansfield.
StackHPC: Stig Telfer

Appendix A


Networking Rack
Spine
Spine-2
Spine-1
Egress
Egress-1
Egress-2
Core
Core-1
Core-1
Rack 2
Hypervisor 2
Vlan trunk
TOR
Leaf-2-1
Leaf-2-2
Lustre Storage 120
lus120
Lustre Top of Rack-1
Lustre Top of Rack-2
Rack 1
Hypervisor 1
Vlan trunk
TOR
Leaf-1-1
Leaf-1-2
Vlan trunk
Lustre Storage 123
OSS6
OSS2
MDS1
OSS1
MDS2
OSS5
OSS3
OSS4
Lustre Top of Rack-1
Lustre Top of Rack-2
Rack 3
HPC 1
Vlan trunk
HPC TOR
Leaf-3-1
Leaf-3-2
Vlan trunk

Detailed network topology.

Each top of rack switch in OpenStack is connected to a spine pair with a single 100GB/s connection to each spine. Any layer 2 network which is stretched over the whole of Openstack is encapsulated into VXLAN on the top of rack switches. Each of the top of rack pairs has a total of 400GB/s of bandwidth to the spines and we have redundancy at every step of the network.

The Egress switch pair are in effect a service top of rack pair, they are the interface between the OpenStack system and the standard HPC environment, today 400GB/s of bandwidth is sufficient between these environments.

The spines are layer 3 routers only by design. The scientific network currently uses OSPF as the underlay routing protocol with iBGP.

Access control lists are applied at all points where packets are routed. Packets exiting OpenStack from standard instances are routed on the egress pair and have an ACL applied to them which is used to protect the HPC systems from arbitrary packets originating from OpenStack. Our policy says that only server-authenticated protocols are allowed access to the HPC systems, this means for example that CIFS is allowed while NFS is not.

Packets originating on provider networks are either routed on the Egress switches where possible but are routed via a distributed router where needed. For our Lustre provider networks we want to allow access to any secure Lustre system. If each Lustre provider network was only ever going to connect to a single Lustre system we could either stretch the VXLAN to each Lustre top of rack router or use the Egress switches to encapsulate and apply ACL's however this is not a general solution.

The more general solution is having each of the Lustre provider networks implemented using a distributed router. On each top of rack switch in the OpenStack system there is a software virtual interface for each of the Lustre networks layer 2 networks (the layer 2 network is stretched over all the switches using VXLAN). The ACL is implemented on each of the top of rack switches. To ensure that traffic is routed to the correct router instance a host route is promoted for each host that is present at layer 2 using "ip attached-host route export" which exports a host route for each connected host via BGP. This solves the problem that traffic for a node in rack1 might be routed to rack2 using Equal Cost Multipath routing, which would then be routed and encapsulated and then switched to rack1 taking an inefficient route.

Our HPC systems are connected to our cores with no ACL's in the path. Our Lustre systems have ACL's which only allow access from either the HPC systems or from the OpenStack Systems. The same ACL could be applied to all the Lustre top of rack switches.

In production the one difference is that lus120 which is used for our faculty production sequencing can only be accessed from OpenStack and our faculty HPC cluster rather than all of the HPC clusters.

Appendix B

Sample top of rack, router configuration snippets

Vlan definitions

First we define each of the layer two vlans, we create 16 vlans per division and allocate them later as projects request access to Lustre.

```
vlan 43
  name lus-hgi01
vlan 44
  name lus-hgi02
```

Software Virtual Interface definitions

Each vlan has a SVI defined. Networks are allocated so that each set of 16 networks per division can be collected together as a supernet /19. The first ip address on the network is allocated as the router interface. We do not run ospf routing on the network. Host routes are created for directly attached devices and these are propagated by BGP. An ACL is applied which only allows traffic to the correct LNet network.

```
interface Vlan43
  mtu 9100
  ip attached-host route export
  ip access-group secure-lustre in
  ip ospf disabled
  ip address virtual 10.177.0.1/23
```

Access control list definitions

This is the ACL which ensures that traffic from the second Lustre only interface can only be routed to the associated LNet network. We allow all established tcp and icmp traffic. Then for each divisional supernet eg 10.177.0.0/19 we have a permit line for each Lustre system (10.160.32.96/27 is lus23, while 10.177.252.96/27 is lus20) we allow access to the Lustre port.

```
ip access-list secure-lustre
  10 remark Established traffic
  20 permit tcp 10.177.0.0/17 10.160.32.0/19 established
  30 remark ICMP Traffic
  40 permit icmp any any
  50 remark Human Genetics
  60 permit tcp 10.177.0.0/19 10.160.32.96/27 eq 988
  70 permit tcp 10.177.0.0/19 10.177.252.96/27 eq 988
  80 remark Human Genetics Deny
  90 deny ip 10.177.0.0/19 any
  100 remark Tree Of Life
  110 permit tcp 10.177.32.0/19 10.160.32.128/27 eq 988
  120 permit tcp 10.177.32.0/19 10.177.252.128/27 eq 988
  130 remark Tree Of Life Deny
  140 deny ip 10.177.32.0/19 any
  300 remark Catchall Deny
  310 deny ip any any log
```

Appendix C

Understanding the hypervisor networking

First we created a whole hypervisor secure lustre instance as this will make debugging the network simpler as the only configuration we should be able to see should be relevant for the guest.

First discover which hypervisor the guest is on

```
# With admin credentials in scope
jb23@farm5-head1:~$ openstack server show -c OS-EXT-SRV-ATTR:instance_name -c
OS-EXT-SRV-ATTR:hypervisor_hostname 5df76968-6841-4e83-812b-0f2c83b0c8ed
+-------------------------------------+-------------------+
| Field                               | Value             |
+-------------------------------------+-------------------+
| OS-EXT-SRV-ATTR:hypervisor_hostname | node-5-7-1        |
| OS-EXT-SRV-ATTR:instance_name       | instance-0004d206 |
+-------------------------------------+-------------------+
```

Login to the hypervisor

```
[root@node-5-7-1 stack]# docker exec -it nova_libvirt /bin/bash
(nova-libvirt)[root@node-5-7-1 /]# virsh dumpxml instance-0004d206
...
   <interface type='bridge'>
     <mac address='fa:16:3e:c1:b8:6b'/>
     <source bridge='qbr81e08ffb-6b'/>
     <target dev='tap81e08ffb-6b'/>
     <model type='virtio'/>
     <mtu size='8950'/>
     <alias name='net0'/>
     <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
   </interface>
   <interface type='bridge'>
     <mac address='fa:16:3e:21:7f:15'/>
     <source bridge='qbr50572119-95'/>
     <target dev='tap50572119-95'/>
     <model type='virtio'/>
     <mtu size='9000'/>
     <alias name='net1'/>
     <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
   </interface>
...
```

Login to the guest

```
ubuntu@jb23manops-full:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8950 qdisc fq_codel state UP group default
qlen 1000
   link/ether fa:16:3e:c1:b8:6b brd ff:ff:ff:ff:ff:ff
   inet 192.168.252.192/24 brd 192.168.252.255 scope global dynamic noprefixroute ens3
      valid_lft 81355sec preferred_lft 81355sec
```

```
   inet6 fe80::f816:3eff:fec1:b86b/64 scope link
      valid_lft forever preferred_lft forever
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default
qlen 1000
   link/ether fa:16:3e:21:7f:15 brd ff:ff:ff:ff:ff:ff
   inet 10.177.129.116/23 brd 10.177.129.255 scope global dynamic noprefixroute ens4
      valid_lft 81355sec preferred_lft 81355sec
   inet6 fe80::f816:3eff:fe21:7f15/64 scope link
      valid_lft forever preferred_lft forever
```

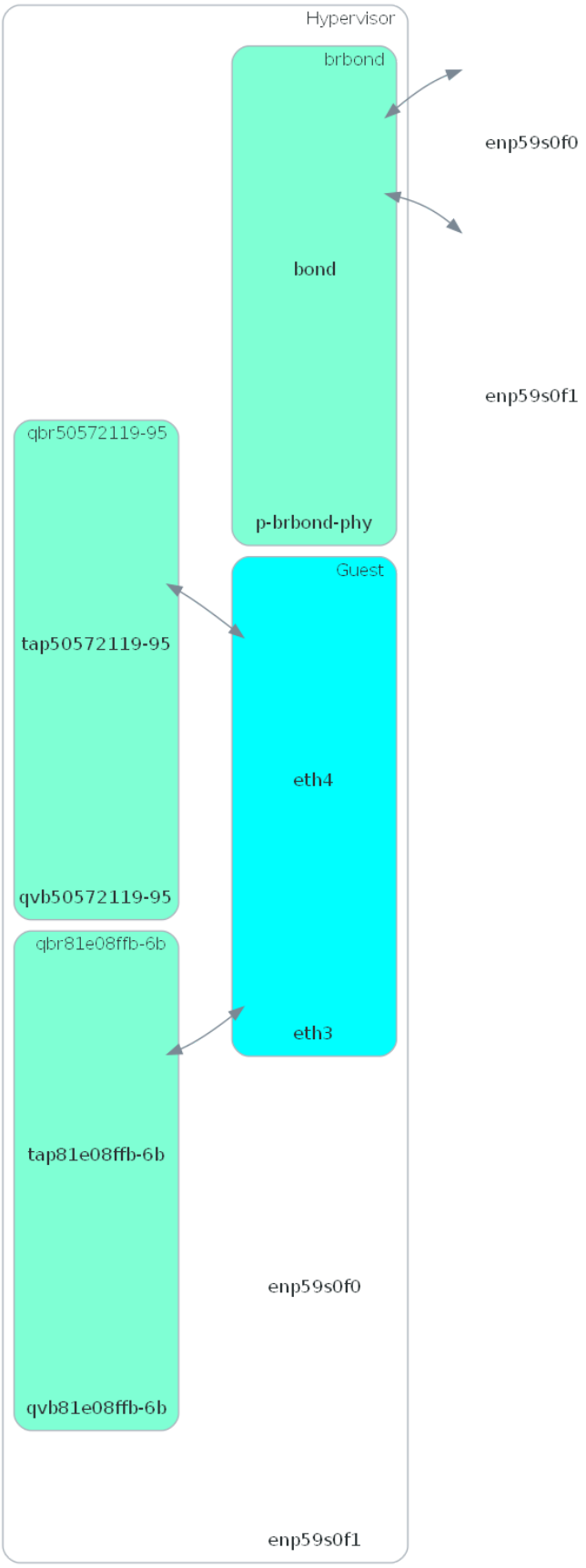
On the hypervisor discover the bridges and the port names on the bridges.

```
[root@node-5-7-1 stack]# docker exec -it nova_compute /bin/bash
(nova-compute)[nova@node-5-7-1 /]$ brctl show
bridge name     bridge id               STP enabled     interfaces
brbond          8000.265a0148bc0b       no              bond
                                                        p-brbond-phy
docker0         8000.024266f15dfa       no
n          8000.868d639b6ff6       no              qvb50572119-95
                                                        tap50572119-95
qbr81e08ffb-6b          8000.f246ab529336       no              qvb81e08ffb-6b
                                                        tap81e08ffb-6b
```

Check the external connectivity

```
[root@node-5-7-1 stack]# ls /proc/net/bonding/
bond
[root@node-5-7-1 stack]# grep "Slave Interface" /proc/net/bonding/*
Slave Interface: enp59s0f1
Slave Interface: enp59s0f0
```

At this point we have this diagram:



Next we need to discover the open vSwitch bridges[6].

```
(openvswitch-vswitchd)[root@node-5-7-1 /]#
ovs-vsctl show
c76c9d4b-311c-446a-b0d2-a509a044ea08
   Manager "ptcp:6640:127.0.0.1"
       is_connected: true
   Bridge brbond-ovs
       Controller "tcp:127.0.0.1:6633"
           is_connected: true
       fail_mode: secure
       datapath_type: system
       Port p-brbond-ovs
           Interface p-brbond-ovs
       Port phy-brbond-ovs
           Interface phy-brbond-ovs
               type: patch
               options: {peer=int-brbond-ovs}
       Port brbond-ovs
           Interface brbond-ovs
               type: internal
   Bridge br-int
       Controller "tcp:127.0.0.1:6633"
           is_connected: true
       fail_mode: secure
       datapath_type: system
       Port int-brbond-ovs
           Interface int-brbond-ovs
               type: patch
               options: {peer=phy-brbond-ovs}
       Port "qvo81e08ffb-6b"
           tag: 33
           Interface "qvo81e08ffb-6b"
       Port br-int
           Interface br-int
               type: internal
       Port patch-tun
           Interface patch-tun
               type: patch
               options: {peer=patch-int}
       Port "qvo50572119-95"
           tag: 32
           Interface "qvo50572119-95"
   Bridge br-tun
       Controller "tcp:127.0.0.1:6633"
           is_connected: true
       fail_mode: secure
       datapath_type: system
       Port patch-int
           Interface patch-int
               type: patch
               options: {peer=patch-tun}
       Port "vxlan-ac1b0aa1"
           Interface "vxlan-ac1b0aa1"
               type: vxlan
               options: {df_default="true",
egress_pkt_mark="0", in_key=flow,
local_ip="172.27.10.74", out_key=flow,
remote_ip="172.27.10.161"}
       Port br-tun
           Interface br-tun
               type: internal
       Port "vxlan-ac1b0a29"
           Interface "vxlan-ac1b0a29"
               type: vxlan
               options: {df_default="true",
egress_pkt_mark="0", in_key=flow,
local_ip="172.27.10.74", out_key=flow,
remote_ip="172.27.10.41"}
       Port "vxlan-ac1b0a2a"
           Interface "vxlan-ac1b0a2a"
               type: vxlan
               options: {df_default="true",
egress_pkt_mark="0", in_key=flow,
local_ip="172.27.10.74", out_key=flow,
remote_ip="172.27.10.42"}
       Port "vxlan-ac1b0a28"
           Interface "vxlan-ac1b0a28"
               type: vxlan
               options: {df_default="true",
egress_pkt_mark="0", in_key=flow,
local_ip="172.27.10.74", out_key=flow,
remote_ip="172.27.10.40"}
```

We have three bridges, brbond-ovs which is connected to the external nic cards, br-int[7] which is the integration bridge and used for all ports connected and br-tun which is responsible for bridging the VXLAN encapsulated mesh of nodes on the hypervisor.

Here port vxlan-ac1b0aa1 is connected to another hypervisor which hosts a second guest on the self service network, ports vxlan-ac1b0a29, vxlan-ac1b0a2a and vxlan-ac1b0a28 connect to the controllers which are responsible for ingress and egress traffic from the self service network.

[6] https://blog.scottlowe.org/2012/11/27/connecting-ovs-bridges-with-patch-ports/
[7] https://superuser.openstack.org/articles/openvswitch-openstack-sdn/

From this we get the following diagram at this point the code which generates the diagram may be more informative.