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 tenant 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 tenants and discuss both physical and virtual Lustre routers. We will examine performance and reliability and discuss using OpenStack virtual machines to extend a conventional batch scheduling system.

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 and others.

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

- Tenant networks are created dynamically and can have any IP address range the user requires, machines provisioned on the tenant 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. GRE or VXLAN.
- 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.
- Provider networks are used by the cloud provider to provide services to tenants and are implemented as VLAN networks. Access to each provider network can be controlled by roles allocated to a user/project.

The current deployment at the Sanger Institute is Red Hat OpenStack Platform 8, which is based on the upstream “Liberty” release. The hardware platform is SuperMicro, with ~5500 CPU cores available to users, and 4PB of usable Ceph storage providing block (Cinder) and S3-compatible (radosgw) access. 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 and controller nodes have 2x100GbE.

Introducing Lustre (for OpenStack users)

Lustre is an open-source high performance POSIX filesystem which has been in use and development 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, Cray, 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@tcp7` and is called a NID, or network identifier. `tcp7` 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.

A Lustre router is a system which has two or more interfaces on different Lustre networks (for example eth0 could be configured as 192.168.11.227@tcp7 while eth1 could be configured as 10.17.17.5@tcp6) and is configured to route packets between its interfaces. Lustre cannot be used though a NAT gateway.

A nodemap[2] is a mapping of a collection of IP addresses 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 tenant25 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.
- 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 64 OSTs, hosted on 8 OSSs.

The Sanger Institute has been using Lustre for many years and the filesystem capacity currently in use exceeds 13PB, provided by DDN. It is entirely ethernet-connected, with each Lustre component server typically having 40GbE 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

Implementation of the Lustre client in OpenStack

There are a number of steps which will need to be repeated for each tenant. In our pilot work, while we have not automated all of these steps, it is clear that they could be automated. However it would be wise to create the provider networks in batches in advance as restarting Neutron (the OpenStack software-defined network controller) can be disruptive. There are a number of areas that need configuration. Before starting, the following information needs to be compiled for each tenant.

Tenant name	UID/GID	Lustre Routers	VLAN for provider network
cpg	32769	172.16.27.35@tcp0, 192.168.11.2@tcp32769	109

By convention we would recommend setting the subdirectory to be the tenant name and the Lustre Network label to be “tcp” followed the UID number, for example “tcp32769” - this makes it easier to follow the configuration and data flow.

Lustre servers

While we require Lustre 2.9 for UID/GID mapping and subdirectory mounts there are a number of reasons to use Lustre 2.10 - not least the many bug fixes[3] - it is a LTS (long term support) release and also adds project quotas and progressive file layouts which are significantly useful incremental improvements. If using Lustre 2.9 then ensure that the fix in LU-9289[4] has been applied.

Create a new directory from an administrative host which can mount the complete filesystem and is trusted.

```
TENANT_NAME=”cpg”
TENANT_UID=32769
mkdir /lustre/secure/${TENANT_NAME}
chown ${TENANT_NAME}  /lustre/secure/${TENANT_NAME}
```

Create a new nodemap:

```
TENANT_NAME=”cpg”
TENANT_UID=32769
lctl nodemap_add ${TENANT_NAME}
lctl nodemap_modify --name ${TENANT_NAME} --property trusted --value 0
lctl nodemap_modify --name ${TENANT_NAME} --property admin --value 0
lctl nodemap_modify --name ${TENANT_NAME} --property squash_uid --value ${TENANT_UID}
lctl nodemap_modify --name ${TENANT_NAME} --property squash_gid --value ${TENANT_UID}
lctl nodemap_add_idmap --name ${TENANT_NAME} --idtype uid --idmap 1000:${TENANT_UID}
lctl set_param -P nodemap.${TENANT_NAME}.fileset=/${TENANT_NAME}
lctl nodemap_add_range --name ${TENANT_NAME} --range \ [0-255].[0-255].[0-255].[0-255]@tcp${TENANT_UID}
```

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 - therefore we set this to false. The “admin” property defines whether root is squashed on the nodemap members, we set it to false in

[3] http://wiki.lustre.org/Release_2.10.0 The lustre 2.10 release notes
[4] https://jira.hpdd.intel.com/browse/LU-9289 Fix fileset string length issue

order to squash root.

We squash all UIDs and GIDs to the reserved UID for this tenant. We explicitly map UID 1000 to the reserved UID because 1000 is the default UID inside many OpenStack images. This makes access more straightforward, in that all files appear to be owned by the default user. An additional benefit is the Lustre quota mechanism then applies nicely to the tenant's filesystem.

Add the routes to the Lustre network to the servers either temporarily[5]

```
lnetctl route add --net tcp${TENANT_UID} --gateway ${LUSTRE_ROUTER_IP}@tcp
```

or persistently[6] via editing `/etc/modprobe.d/lustre.conf` .

```
options lnet networks="tcp32769" routes="tcp0 172.16.27.35@tcp0"
```

Lustre routers

Lustre routers should run the same version of Lustre as that the clients and the servers. First configure each of the network interfaces and ensure that IP connectivity is working. The most relevant file is `/etc/modprobe.d/lustre.conf` which configures the binding of Lustre networks to network interfaces. Specific configuration options are described in the Lustre operations manual; a typical configuration in our prototype for a filesystem on tcp0 (VLAN 12) and a tenant on tcp32769 (VLAN 109) is:

```
options lnet networks="tcp0(bond0.12),tcp32769(bond0.109)"
```

The Lustre operations manual contains several relevant areas including configuring router buffers [7][8], the router checker[9] and router resilience[10] - however these subjects are beyond the scope of this document.

Lustre client configuration

Lustre client configuration relies on two files: `/etc/modprobe.d/lustre.conf` to configure the Lustre software kernel modules:

```
options ptlrpc at_min=100 at_max=1200
lnet networks="tcp32769(eth0)" routes="tcp0 172.16.27.35@tcp32769"
```

and `/etc/fstab` to configure which filesystems are mounted at boot:

```
lus-mds1@tcp0:lus-mds2@tcp0:/lus07    /lustre/secure   lustre   localflock,noauto    0 0
```

[5] http://doc.lustre.org/lustre_manual.xhtml#idm140684903155904 Adding, deleting and showing routes.
[6] http://doc.lustre.org/lustre_manual.xhtml#idm140684903001584 Routing examples
[7] https://www.eofs.eu/_media/events/lad15/15_chris_horn_lad_2015_lnet.pdf LNet and LND tuning
[8] http://doc.lustre.org/lustre_manual.xhtml#idm140684903138880 9.1.7 Router buffers
[9] http://doc.lustre.org/lustre_manual.xhtml#dbdoclet.50438216_35668 9.7 The router checker
[10] http://doc.lustre.org/lustre_manual.xhtml#dbdoclet.mrroutingresiliency 16.3.2 Router resilience

OpenStack network, role and policy configuration

The first step is to create a layer 2 VLAN that is available to all hypervisors. As an OpenStack admin a shared provider network is created, specifying the VLAN number.

```
neutron net-create <name> --shared --provider:network_type vlan  \
--provider:physical_network datacentre --provider:segmentation_id <VLAN number>
```

As a OpenStack admin we create a subnet and configure the DHCP service that is provided.

```
neutron subnet-create  --enable-dhcp --dns-nameserver 172.18.255.1 \
--dns-nameserver 172.18.255.2 --dns-nameserver 172.18.255.3  --no-gateway --name LNet-subnet-1 \
--allocation-pool  start=172.27.202.17,end=172.27.203.240 <network UUID> <CIDR>
```

Next we create a role per tenant that is using Lustre:

```
openstack role create <role name>
```

Then we can assign that role to a user+tenant combination, so that accounts which have that role will be able to provision instances that can have a NIC on the Lustre network. By being able to have a NIC on the network the machine is granted access to the Lustre area and anyone who has access to the instance has access to the data.

```
openstack role add --project <project ID> --user <user ID> <roleID>
```

Here we change the default rule in `/etc/neutron/policy.json` which controls whether a user/tenant pair can access a network. At Sanger this is implemented with a minimal change to one existing configuration file and by creating two new files; we do this to simplify the Ansible automation used to make the change.

The existing Neutron policy file `/etc/neutron/policy.json` is modified to contain:

```
"get_network": "rule:get_network_local"
```

The new file `/etc/neutron/policy.d/get_networks_local.json` holds our new “get_network_local” rule which the above file references. This keeps the change to `/etc/neutron/policy.json` simple.

```
{
"get_network_local": "rule:admin_or_owner or rule:external or rule:context_is_advsvc or
rule:show_providers or ( not rule:provider_networks and rule:shared )"
}
```

We define a rule which matches the network created earlier and a rule which allows access to that network if you have the correct rule.

The new file `/etc/neutron/policy.d/provider.json` is used to define networks and their mapping to roles. For example, this defines policy for two tenants and defines the provider_networks and show_provider rules which are used in `get_networks_local.json`

```
{
        "net_LNet-1": "field:networks:id=d18f2aca-163b-4fc7-a493-237e383c1aa9",
        "show_LNet-1": "rule:net_LNet-1 and role:LNet-1_ok",
        "net_LNet-2": "field:networks:id=169b54c9-4292-478b-ac72-272725a26263",
        "show_LNet-2": "rule:net_LNet-2 and role:LNet-2_ok",
        "provider_networks": "rule:net_LNet-1 or rule:net_LNet-2",
        "show_providers":      "rule:show_LNet-1 or rule:show_LNet-2"
}
```

After all the rules are changed Neutron needs to be restarted on the controllers.

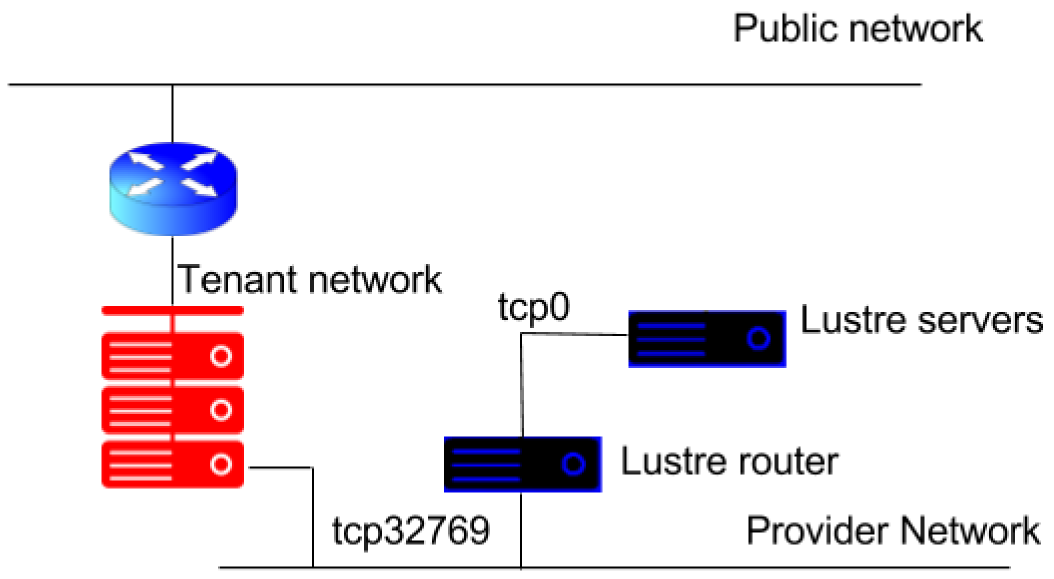
Comparing physical and virtual Lustre routers

Given the acceptable performance of bare metal shared Lustre routers we wanted to discover if we could virtualise the Lustre routers. This would allow us to scale the number of routers to provide additional redundancy and performance, while also isolating Lustre router issues to a single tenant. This would also be a cost effective solution for convenience (where the ability to use the filesystem is more important than performance).

We expected to have performance implications with significantly more east/west traffic generated, however our network was designed to allow more east/west traffic than the layouts used for our traditional high throughput clusters.

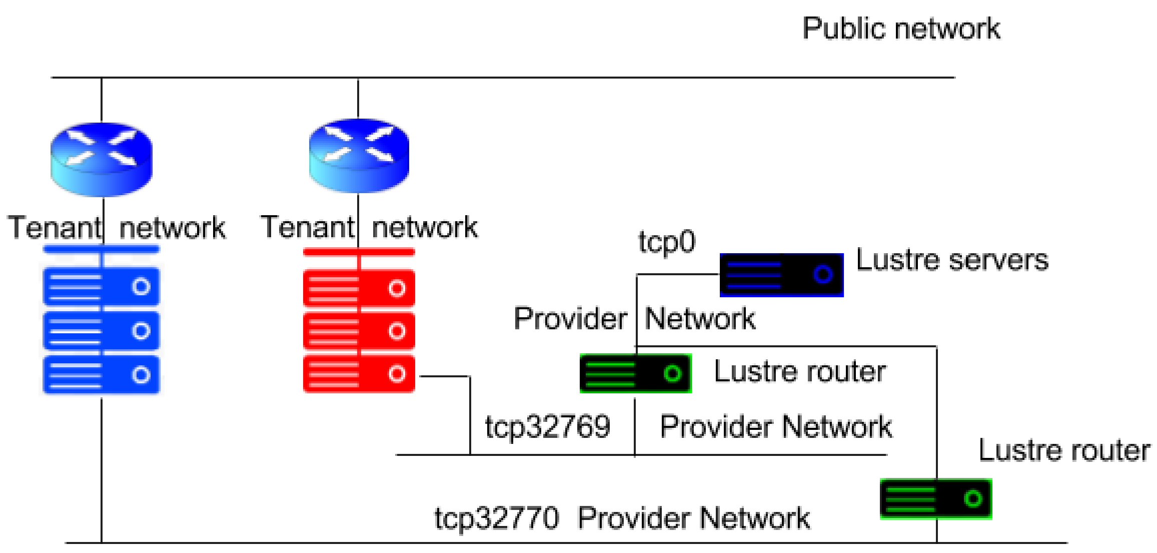
As each tenant has a separate Lustre router even if the Lustre router was compromised then one would have to use the compromised machine as a stepping stone to either preferably attack the Lustre servers or other tenants' Lustre routers. Given that the attack surface should still only be the Lustre port (988) this would be a relatively complex attack.

Topology with physical Lustre router



Red machines are instances belonging to tenant 32769, while blue-black machines are physical.

Topology with virtual Lustre routers



Red machines are instances belonging to tenant 32769, blue machines are instances belonging to tenant 32770, green-black machines are instances belonging to the Lustre-router tenant, while blue-black machines are physical.

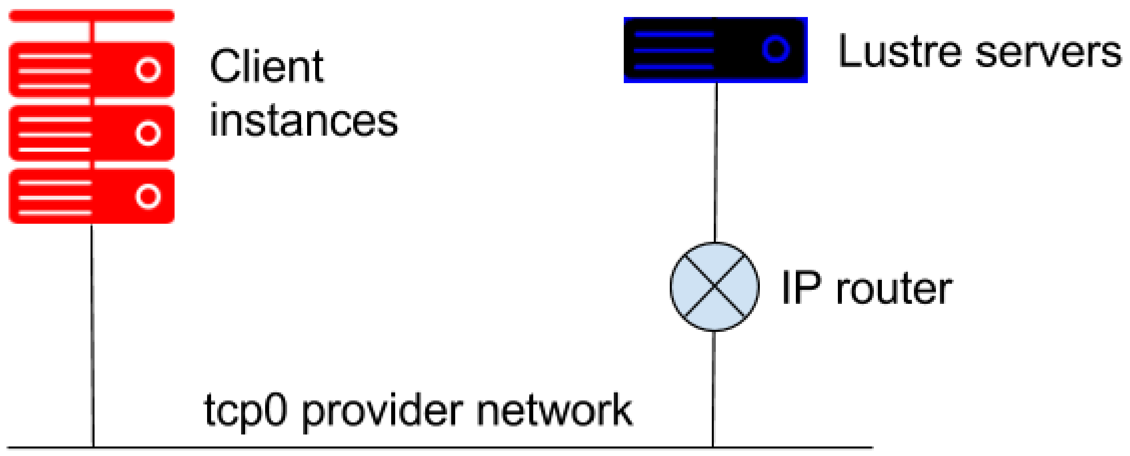
When the virtual Lustre routers were provisioned and given access to the shared tenant network the virtual Lustre routers initially couldn't be accessed from the tenant instances. After some investigation it was found that disabling port security for the virtual Lustre routers was necessary[11]. This is acceptable to us because the Lustre filesystem, Lustre routers and OpenStack infrastructure are all under the same administrative domain. It would be considered sufficient to configure iptables within the Lustre router instance to allow only TCP/988 from the tenant side.

```
neutron port-list | grep 172.27.70.36 | awk '{print $2}'
08a1808a-fe4a-463c-b755-397aedd0b36c
neutron port-update --no-security-groups 08a1808a-fe4a-463c-b755-397aedd0b36c
neutron port-update 08a1808a-fe4a-463c-b755-397aedd0b36c \ --port-security-enabled=False
```

Extending standard batch queuing systems

The reason for for using Lustre routers in the examples so far is to isolate access (and therefore restrict access to data) because the users of the client systems - OpenStack instances - have access to a privileged account (root) and so could bypass Lustre's POSIX security model. If however we consider an environment where the administrator of the OpenStack instances is the same as the administrator of the filesystem, and the users do not have root access, then we can connect the clients to the filesystem without any restrictions. A physical Lustre filesystem cannot be accessed by OpenStack instances connected to a conventional tenant network, because their traffic is NAT'ed at the tenant network router. This breaks Lustre's assumption that the servers can contact the clients, i.e. that the clients' IP addresses are routable. Instead we can use a provider network to allow access to the filesystem. The Lustre servers do not need to be attached to the provider network, standard IP routing is sufficient.

[11] https://bugs.launchpad.net/nova/+bug/1554728 Unable to launch instance on a network where port-security-enabled=False



We have used this technique to prototype adding virtual compute nodes to an LSF cluster alongside physical compute nodes. The users see identical access to the Lustre filesystems, and the virtual compute nodes are managed and configured by Ansible in exactly the same way as the physical compute nodes.

Reliability

Client failover between multiple Lustre routers is an established technique for providing high availability. We have prototyped this with both physical and virtual Lustre routers. The client can be configured with the addresses of the various routers it may use in `/etc/modprobe.d/lustre.conf`

```
options lnet networks="tcp32769" routes="tcp0 172.16.27.3[1-5]@tcp0"
```

LNet routing is discussed in detail in the Lustre Operations Manual[12].

Performance

Given the constraints of the decommissioned Lustre array available to this proof of concept, the performance figures were satisfactory. A single physical client achieved ~5GB/s random writes, and a single virtual client accessing the filesystem via Lustre routing was able to exceed 2GB/s.

The biggest contribution to improving performance was found to be upgrading the hypervisor’s kernel to gain a bugfix in the Mellanox network driver. This improved instance networking performance by a factor of two.

All LNet routers that bridge two networks are equivalent. They are not configured as primary or secondary, and the load is balanced across all available routers. Therefore it is possible to scale up the number of routers to achieve a desired performance target. In early cursory testing, we found diminishing returns when adding routers:

[12] http://doc.lustre.org/lustre_manual.xhtml#configuringlnet

Number of virtual routers	Throughput (MB/s)
1	502
2	820
4	1054

Future work

LSF provides the facility for a cluster to have nodes added or removed dynamically[13] - that is, without the disruption of restarting the master batch daemons which is necessary for a conventional cluster reconfiguration. In combination with the technique of using a provider network to connect Lustre clients, this enables the possibility of adding virtual capacity to a compute cluster, either as part of a physical-to-virtual transition, or as a way of temporarily adding compute capacity on-demand. It would be necessary to develop some method of monitoring the number of jobs pending in one or more queues to trigger adding instances; and similarly monitoring the number of idle instances, to trigger removing instances.

There are also many opportunities for performance tuning which by and large took a back seat during the implementation of this proof of concept, for example increasing the MTU.

Conclusion

We have satisfactorily used the combination of OpenStack's provider networks, Neutron policy configuration and Lustre routers to provide secure, isolated multi-tenant access from user-controlled instances to Lustre filesystems. In addition, where the instances are not under user control, a provider network can be used to connect instances, such that the user experience is identical on physical and virtual Lustre client systems. This provides a way forward for seamless transition from physical clusters to virtual clusters, or for on-demand capacity bursting by adding virtual compute nodes to a physical cluster.

Acknowledgements

Sanger Institute - James Beal, Jonathan Nicholson, Dave Holland, Peter Clapham, Matthew Venon, Tim Cutts.
DDN - Sébastien Buisson, Richard Mansfield, James Coomer.

[13] See, for example, https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.2/lsf_admin/dynamic_hosts_lsf.html