Network Integration of Perlmutter at NERSC

Ershaad Ahamed Basheer , Eric Roman, Tavia Stone Gibbins, Christopher Samuel, Lisa Gerhardt, Ashwin Selvarajan, Damian Hazen, Douglas M. Jacobsen, Ronal Kumar
Lawrence Berkeley National Laboratory
{ebasheer, eroman, tavia, csamuel, lgerhardt, dmjacobsen, apselvarajan, dhazen,, ron}@lbl.gov

***Abstract*— In this paper, we describe the integration of NERSC's HPE Cray EX supercomputer "Perlmutter" into the NERSC data center. Perlmutter connects to NERSC via 4 networks. The Customer Access Network (CAN) provides administrative access to the API gateways; Users access the 40 login nodes over the high-speed network (HSN) via an internal two-tier load balancer. A second external management network provides access to the first master node and SMNet out-of-band switch ports. GPFS is IP-routed through 24 gateway nodes to a separate Infiniband fabric. The Slingshot network connects to the site network via a 64-way LAG. Since every node in the cluster has a routable IP address, nodes operate directly with other NERSC resources over the high-bandwidth Ethernet fabric. We show that the resulting configuration is (1) secure because it allows us to isolate the administrative networks via routing rules in the data center routers, (2) reliable because administrators can access the NCNs and switches independently of the availability of the SMNet or HSN, and (3) performant because all high-bandwidth traffic is confined to the HSN via edge routers. As a result, NERSC has been able to make Perlmutter into a reliable high-performance system for our users.**

Keywords—networking, data center integration, smnet, hsn, hpe-cray ex, CAN, load balancing

I. Privatized Customer Access Network

The Shasta software stack uses secure tokens exchanged with an API gateway to manage administrative access to the system. While this can be an effective approach, NERSC's security policies require that administrative access be restricted to MFA-authenticated networks. To implement this policy, we decided to restrict users from accessing the CAN entirely. All user traffic is routed through the edge routers to the Slingshot network. Access control lists enforced in the data center routers block the CAN IP range from all but the MFA-secured site networks. CAN traffic is generally limited to a few services, e.g. NTP, DNS, and SSH, within the data center. Access to the API gateway is limited to a dedicated management virtual machine (VM), which is available even if the Shasta system is completely down, and serves as the single place where all system administration tasks and management workflows are run. Administrators run administrative tools such as the cray command, kubectl and sat, and other Shasta services from the management VM. Our administrators use SSH to the non-compute nodes (NCNs) only in cases where no suitable interface is available.

The standard Shasta software stack installation uplinks the CAN to the data center networks via a static default route configured on the SMNet spine switches. For increased reliability, we chose to employ point-to-point routes between the two SMNet spines and our data center routers as shown in Figure 1. The SMNet and data center routers exchange routes via OSPF, where the SMNet spines are configured as a separate not-so-stubby area. The data center routers export a summarized route to the SMNet routers. To avoid the possibility of exporting the SMNet RFC 1918 routes to the data center, e.g. the 10.252.0.0/17 NMN, we chose to use a separate CAN VRF for the data center routes. The use of an independent VRF required BGP configuration changes. First, metalLB needed to be adjusted to peer, in addition to the default VRF, on the new CAN VRF as well. Corresponding changes needed to be introduced on the SMNet router for the additional peering via CAN, and route maps were added to match CAN IP prefixes. Ultimately we had a configuration that allowed our SMNet and data center routers to be managed largely independently, provided configuration flexibility on both the data center and SMNet routers, and reliability due in large part to the fast convergence of OSPF.

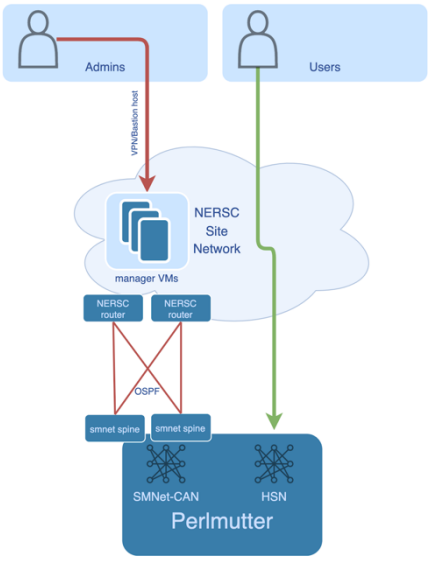

Figure 1 Privatized CAN network topology

II. External Management Network

Although the CAN network is the primary administrative route into a Shasta system, we needed a secondary external management network to be available when the CAN is down. This occurs during the early phases of a CSM installation, which requires that the baseboard management controller (BMC) of the first master node (m001) be connected to a site network. Even before the CSM installation can proceed on the first master node, the SMNet switches must be configured. We added the external management network to address these needs. This network connects the manager virtual machine with the BMC of the first master node, the management ports of all SMNet switches, and serial console servers (Figure 2). The serial console servers connect to the serial ports of the SMNet switches, and to the console ports of the River Rosetta switches in our development systems. Unlike the CAN, which retains limited routability within the data center, the external management network is an isolated layer-2 segment on a private RFC 1918 IP range. This network is only accessible from the manager virtual machine. Beyond the initial bootstrap, we have found this network valuable during regular operations too. Out-of-band access to the SMNet routers provides us a level of resilience to configuration errors in the SMNet, and has given us more freedom to take risks when managing the SMNet configurations. In addition, it provides a conduit for system logs, allows us to perform firmware updates, and collect debug data when needed by vendor support. Although the external management network is not a part of day-to-day operations, we have found it to be an invaluable resource during major maintenance efforts.

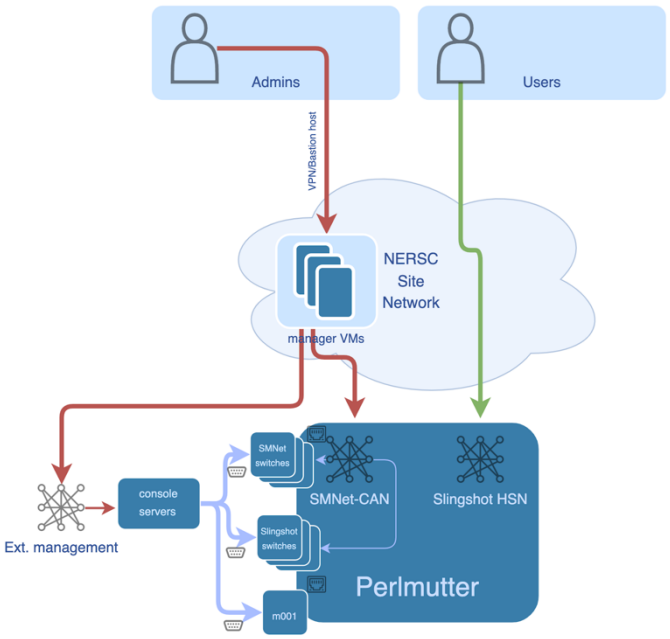

Figure 2 External Management Network

III. CSM external DNS service

The default CSM external DNS service is currently provided by a coredns deployment which serves records read from etcd, which in turn are populated by CSM, rather than the traditional zone file arrangement used for many DNS services. The coredns external DNS service runs as a deployment within the CSM Kubernetes cluster and the endpoint (a static IPv4 address) is accessed via the Customer Access Network (CAN). This arrangement presents some immediate challenges for production use.

- The mechanism used to provide the CAN only supports IPv4 and so IPv6 services are not possible.
- The etcd plugin for coredns does not support AXFR or IXFR zone transfers, and so it is not possible to have a secondary DNS server of any form, which is a requirement for any best practice DNS deployment.
- For delegations in .gov the use of DNSSEC is mandatory and there was no provision for supporting that with the default configuration.

To get an initial functional deployment in place to allow installation and bring up despite these limitations it was agreed with LBL & NERSC that they would delegate domains under nersc.gov to the systems, but that NERSC would ensure that DNS queries to those systems were only possible from inside the NERSC boundary. In order to make the login nodes accessible to users some static records for the load balanced IPs for them were put directly into the nersc.gov domain.

With advice from HPE we were able to configure a simple DNSSEC configuration by generating appropriate DNSSEC keys on the manager VM and then loading them into the NERSC vault on that VM. We then scripted the creation of Kubernetes secrets from that vault and modified the coredns external DNS deployment to make them available securely within the coredns container. The final step was to modify the Kubernetes configmap for the coredns service to make it aware of those keys and to enable DNSSEC for both forward and reverse lookups. We were then able to communicate those keys to LBL IT and add it to the delegation.

Whilst having working DNSSEC is essential it is not sufficient to enable the exposure of the systems external DNS service to the outside world, and so we have been investigating with HPE ways to achieve this. There are two main options currently under investigation:

A. Dynamically update an external DNS service

Use a separate service of some form that will use nsupdate from BIND to send dynamic DNS updates to an LBL DNS server which can then serve out that information on our behalf to external users.

- This has some significant advantages:
- LBL would handle all DNSSEC duties for us
- LBL DNS servers already have functional IPv6
- There are existing secondary servers in place for resilience
- This does not rely on changes in CSM.

This approach does have some drawbacks though:

- Having the data stored in etcd means we can't know when changes have been made, so it would be necessary to poll and compare with the public DNS to determine what changes need to be made, This would introduce some measure of latency between a change occurring and public records being made available, which would not occur with an on-system DNS server.
- In order to determine whether any records have been left stranded in the upstream DNS due to this service not being able to notify them of their withdrawal, it would be necessary to obtain a zone transfer of the zone we are monitoring and compare it to the information stored internally and ensure the public-facing DNS is up to date.

An initial script to map out this as a way forward was created by HPE and is the basis of current investigations on how to extend it to meet our needs.

B. Migrate to a DNS service that supports zone transfers

Prior to the idea of using dynamic updates to LBL DNS servers, the thought was to use DNS zone transfers as a way of providing the full zone to LBL so they could act as the public face of DNS for the system. This was attractive because it provides a standard, simple way to get the information upstream but the initial investigation was tripped up when the aforementioned fact that the etcd plugin for coredns does not support this mechanism at all.

Advantages to this approach are:

- A consistent view of the zone - every update will pick up the full state at that point in time
- Notifications to the secondary servers should be possible when the zone is updated, allowing minimal delay in updates (modulo the usual concerns around long DNS Time To Live settings on resource records)
- LBL DNS servers already have functional IPv6
- There are existing secondary servers in place for resilience

The disadvantages are the inverse of some of the advantages of the dynamic DNS approach:

- This requires support in CSM to be possible
- We would need to handle all DNSSEC duties and would need CSM to provide reliable support for things like key rotation and separate zone and key signing key support

IV. Storage Network Gateways

A. Site Infrastructure

Perlmutter has access to multiple tiers of storage. The most tightly integrated storage tier is the all-flash-based Lustre filesystem. The second tier comprises the large-capacity GPFS filesystems maintained by NERSC's Storage Systems Group, and the third is long-term tape archival storage. The GPFS filesystems, which are shared across all systems at NERSC, provide the Community File System and users' home directories. The GPFS cluster and all systems that mount it directly are interconnected by an FDR Infiniband fabric which, in conjunction with GPFS support for RDMA, allows high-speed access to the GPFS filesystem. RDMA makes this possible by minimizing the involvement of the host CPU in data transfer between the HCA and the user application, and also by utilizing GPFS's ability to balance load across multiple Infiniband links on the same host.

On our Cray XC systems, the Data Virtualization Service (DVS) distributes cluster filesystems such as GPFS to a large number of compute nodes. DVS is an I/O forwarding layer, which designates a relatively small number of nodes as DVS servers that directly mount the cluster filesystem, and which perform I/O operations that have been forwarded from a much larger set of compute nodes, thereby effectively reducing the maximum number of nodes that directly mount the cluster filesystem. By funneling I/O to the DVS servers and thereby limiting the degree of scale-out of the cluster filesystem, the overhead and complexity of maintaining a coherent distributed filesystem state are kept under control.

B. Native GPFS on Perlmutter

With the continuous improvements in GPFS scaling and the arrival of Ethernet compatible Slingshot, natively mounting GPFS on each and every compute node has become a practical option. Natively mounting GPFS has some desirable advantages including the ability to use the full suite of GPFS management and diagnostic tools and commands. It is additionally attractive since a future upgrade of the IB storage fabric to Ethernet will mean that compute nodes can route directly to the GPFS servers without requiring any protocol gateways, and can even support native RDMA with RoCE.

As illustrated in Figure 3 Perlmutter has 24 service nodes that act as gateways between the Slingshot and Infiniband networks. Since there is no support for RDMA on compute nodes, the gateways simply route IP packets between the Ethernet-compatible Slingshot network and the Infiniband storage network. IP over Infiniband (IPoIB) encapsulation is used to overlay an IP network over the Infiniband fabric. The gateway servers are Linux servers that have connectivity to both Slingshot and Infiniband and are configured to forward IP traffic. Our primary requirements with the architecture for gateways were:

- Resiliency against failures of gateway nodes
- Balanced network traffic across all the gateways

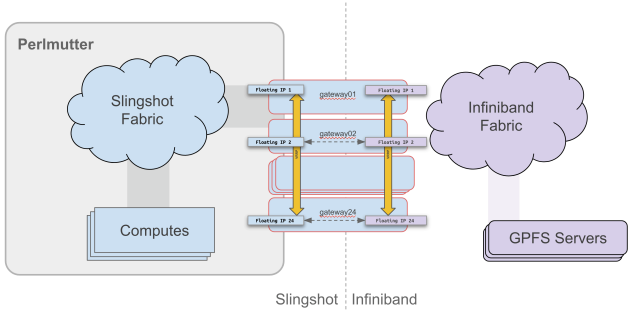

Figure 3 GPFS Storage Gateways

C. Design and implementation

In order to fulfill these requirements, the first design goal was to prevent any compute node from losing connectivity when one or more gateways fail or lose network connectivity. From the point of view of a compute node, a gateway is simply a next-hop IP address for traffic that is destined to servers on the Infiniband fabric. In order for a compute node to have uninterrupted connectivity, we need to make sure any network packets that are sent to a gateway IP address will be forwarded to their destination even if the specific gateway that the IP address belonged to goes offline. We achieve this by assigning each gateway server a Virtual or Floating IP address (VIP) in addition to any static IP addresses on both the Slingshot and Infiniband facing network interfaces. By ensuring that the VIP of a gateway that goes offline is taken over by an online gateway, clients will transparently send packets to a working gateway server if one or more of them go offline. Also when an offline gateway comes back online, its VIP is automatically restored to it, and packets are forwarded again like before. Thus any load-balancing across gateways is only temporarily reduced by the loss of a gateway until it is restored.

The automatic transfer of VIPs between gateways is achieved completely in software by using the open-source implementation of the Virtual Router Redundancy Protocol, or VRRP (RFC 2338) called Keepalived. In general, Keepalived uses IP multicast so that all the routers in a redundancy group can monitor the availability of the primary server. When routers detect the loss of the primary server, an election is triggered and a new primary server is chosen. The new primary server is then configured with the VIP of the failed server and advertises this on the network by broadcasting gratuitous ARP requests on the network.

We configured Keepalived such that we have one redundancy group defined for every VIP, each of these redundancy groups consisting of all 24 gateways. By setting up each of the groups to contain all 24 servers instead of just a pair, the VIP of a server can be taken over by any server that has sufficient priority. For example, if server Ⓐ goes offline, its VIP is taken over by server Ⓑ, presuming Ⓑ has been configured to have the highest priority for that VIP after Ⓐ. Now, if server Ⓑ also goes offline, server Ⓒ will have the next highest priority and will now take over the VIPs of both server Ⓐ and Ⓑ. This can continue until there is only one server online, at which point it has all 24 VIPs configured on it. In our real-world configuration, priorities are assigned randomly between servers for each of the redundancy groups so that VIPs don't bunch up on specific nodes. So, for example, in the scenario above where both servers Ⓐ and Ⓑ are offline, the VIP of server Ⓐ may be assigned to server Ⓒ, while the VIP of server Ⓑ is assigned to server Ⓖ, for instance. This keeps the VIPs, and therefore the network load, spread out as much as possible across all the servers that are still online at any given time.

While VRRP and Keepalived solve the issue of resiliency to gateway failures by ensuring that all 24 VIPs are active as long as at least one gateway server is online, it only provides one part of the solution that we designed. The other half is on the compute nodes. There are many approaches to configuring the next-hop gateway VIPs in the routing tables on the compute nodes. Two are, dynamic assignment managed by a routing daemon and static assignment of a VIP to a compute node in order to distribute load across the gateways. But a relatively new feature in the Linux kernel networking stack provides an elegant way to achieve this: ECMP. Equal Cost Multi-Pathing allows us to define more than one next-hop IP address for a single routing table entry in Linux. When the networking stack encounters a packet that matches this routing table entry, it chooses a next-hop address based on a hash of the headers of the packet. Sysctls in the /proc filesystem allows us to tune this behavior. We configured all the compute nodes on Perlmutter with the same route that lists all 24 gateway VIPs as next-hops to reach the GPFS servers and tuned the sysctls so that both L3 (IP layer) and L4 (TCP layer) are used to calculate the hash that determines the next-hop that is used. L3+L4 hashing, which uses the tuple consisting of source IP address, destination IP address, protocol, source TCP port, and destination TCP port, ensures that all the packets that are part of the same TCP stream always use the same next-hop. This avoids the possibility of packet reordering if packets from the same stream took different paths, which may lead to a loss of performance.

D. Performance and Conclusions

Figure 4 shows a comparison of the performance of native GPFS vs. DVS projection on Perlmutter.

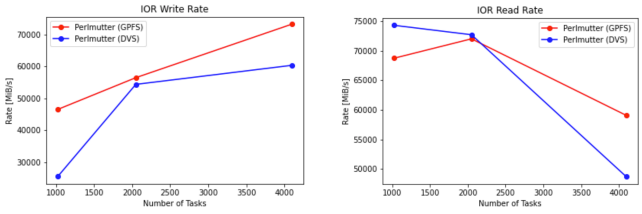

Figure 4 Native GPFS vs. DVS Performance

While native GPFS mounts were not expected to outperform DVS, primarily due to the inability to utilize both Infiniband interfaces available on service nodes, these early benchmark results indicated a lower-than-expected DVS performance. Our hypothesis is that this is due to congestion on the network caused due to fan-in traffic from various systems ingressing into the same fabric Line Card. While these benchmarks are not conclusive and there are ongoing tuning efforts, we have found that native GPFS performance has been

generally good and doesn't degrade the user experience. This is despite the fact that native mounts cannot take advantage of the performance benefits of RDMA.

V. Load-balancing login SSH sessions

Most users of the Perlmutter supercomputer access it remotely using SSH which gives them a shell on one of the 40 login node UANs. A load balancing setup is required in order to distribute the users' shell sessions across all the login nodes more or less uniformly. On other NERSC systems like Cori, this is handled by a dedicated hardware load-balancer, which has disadvantages, including high procurement cost and being a bottleneck for ingress network traffic. The essential requirements for the design of a load-balancing system for Perlmutter were the following

- Balance SSH login sessions across 40 login nodes
- Utilize all 40 nodes for ingress traffic. In contrast with Cori which has one dedicated load balancer that handles all ingress traffic.
- Minimize disruption of established SSH sessions if nodes fail

For SSH session load-balancing, we are able to base the solution on the same basic technique of using floating virtual IP addresses that are in use for gateways nodes. The complete setup for distributing sessions among login nodes is comprised of the 3 components described below.

A. DNS

All 40 login nodes in Perlmutter are configured with an identical user environment, so a user that initiates a connection to Perlmutter can be directed to any of the login servers where they will have access to their home directories on the shared global filesystem and the ability to submit jobs to the batch scheduler. As far as the view from the public Internet is concerned, load-balancing of SSH access to login nodes is handled in a similar way to how it is handled for many popular web-based services and websites, where a single DNS address resolves to multiple IP addresses. The DNS hostname that users use to login to Perlmutter is login.perlmutter.nersc.gov which resolves to 40 IP addresses belonging to each of the login nodes. The order in which these addresses are returned by the DNS server is randomized and therefore each client will generally pick a different IP address and therefore end up on different login nodes.

While this form of load-balancing is conceptually simple, it presents two limitations for our use case. First, there is no quick way to take a login node out of the set of load-balanced IP addresses if, e.g., a login node needs to be taken offline for maintenance. Clients generally cache DNS entries and therefore dynamically updating the DNS entry may not be effective and is very dependent on client configuration. Second, when faulty login nodes establish connections slowly or take a long time to drop a connection, users may be unable to login to an otherwise working login node. It's also possible in this case that the connection process becomes very slow as the client tries each IP address in succession.

B. Virtual Ips

These limitations can be avoided by guaranteeing that IP addresses available to clients are assigned to login nodes that are both healthy and accepting connections. We achieve this in the same way as the gateway solution, where we assign a VIP to each of the login nodes in addition to any existing static IP addresses. Keepalived detects when a login node goes offline and migrates its VIP to a working server. In this way, even if only one login node out of the 40 is online, a client connecting to any of the VIPs will be directed to this online node. This was our initial solution during the early stages of system deployment, but there is an obvious problem with this approach alone, especially since SSH is based on persistent TCP connections. Consider the following sequence of events. A login node Ⓐ with IP address VIP1 goes offline for any reason and Keepalived migrates VIP1 to another login node Ⓑ. Following this, a client establishes an SSH connection to VIP1, now assigned to Ⓑ. Now, while this connection is active, if Ⓐ comes back online, Keepalived will reassign VIP1 back to Ⓐ. Once this happens network packets sent to VIP1 are directed to Ⓐ, even though the connection was originally established on Ⓑ. Since Ⓐ does not recognize this packet as belonging to any established socket or TCP connection, it replies with an RST, causing the connection to be dropped immediately and the user's session to be disconnected. The solution we found to this problem was to introduce an additional level of indirection and IPVS was found to fit this requirement.

C. IPVS

IP Virtual Server (IPVS), which is part of the Linux Virtual Server (LVS) project implements transport-layer load balancing inside the Linux kernel. IPVS running on a host acts as a load balancer in front of a cluster of backend servers called Real Servers. It can direct requests for TCP/UDP-based services to the Real Servers based on configurable criteria and makes services replicated by multiple Real Servers appear as a single Virtual Service available via a single IP address. On Perlmutter, each login node is configured to expose one virtual service for each of the 40 VIPs. The real servers that serve as the back-ends for each of these VIPs consist of the static/fixed IP addresses of all the login nodes. IPVS is configured with the Source Hashing scheduler with the sh-port option enabled. What this means is that when a client establishes a connection to a VIP, which is the Virtual Service IP, decides which backend real server that connection should be redirected to based on the source IP address and source TCP port of the incoming connection. We configure the IPVS Virtual Service to Real Server mapping to have Real Servers in exactly the same order on all the login nodes and this is key to solving the problem with VIPs alone described above. Although Source Hashing is not a consistent hashing algorithm in the same way that the Rendezvous and Maglev algorithms, to name a few, we do not require such an algorithm in our use case since the number of Real Servers is static and does not shrink or grow.

The mechanism by which IPVS solves the problem of clients getting disconnected when a VIP is restored to a login node that was offline is illustrated by the following sequence of events as illustrated in Figure 5:

a) Round-robin DNS directs client sessions across different Virtual IPs. An IPVS Hash function takes the client's IP address and port number as input. It then decides which node the connection is redirected to. Every node is initialized with identical hash tables. In the figure below, the client session has the destination IP of the VIP/Virtual Service on login01. The packets are intercepted by IPVS and then routed to static/permanent IP address of login03 based on the hash table. The SSH client eventually connects to the SSH server on login03.

b) New connections from clients may be directed to a different VIP, but thanks to the identical hash tables on each of the login nodes, the connection ends up on the same server, determined by the hash function, regardless of which VIP it arrives on.

c) In the event of a server node failure, the VIP of the failed node is migrated to an online node. Network packets from already established connections to this VIP now arrive on the new node. Since the hash calculation is identical to what was on the failed node, the packets are correctly routed to this new node. Therefore, except for those on the failed node, all sessions remain uninterrupted.

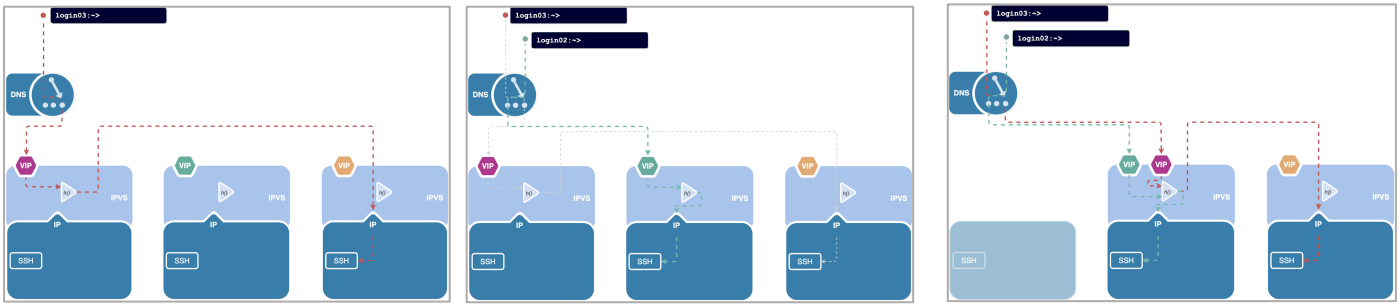

Figure 5 VRRP and IPVS login load balancing

D. Health Checks for Real Servers

While it is possible to manage the IPVS configuration independently of Keepalived using configuration management, one of Keepalived's powerful features is the ability to manage IPVS configuration automatically. By managing IPVS configuration dynamically, Keepalived can keep it in sync with the changing state of the topology of the servers participating in the VRRP group. For example, Keepalived can set the load-balance weight for a Real Server down to zero based on the exit status of a script to prevent IPVS from routing new sessions to those servers. This allows us to dynamically remove login nodes from the load-balance group based on the result of health check scripts that are run at regular intervals. The health check scripts check the health of filesystems, network interfaces, and the presence of administrative files that allow us to drain a node for maintenance, thereby preventing new sessions from landing on the node.

VI. SLINGSHOT HSN

One of the most exciting features of Slingshot is its Ethernet compatibility. The Ethernet compatible high-performance interconnect enables site-integration of the Perlmutter system to data center networks and beyond. As a result, compute nodes connect directly with site-hosted, and externally hosted, services and data sources. To make this final design a reality, many hardware and software configurations were put into place to ensure high-performance, resilient routing between the Perlmutter Slingshot interconnect and NERSC's site networks.

We deployed a pair of Arista 7808 chassis to support our LAG connections to the Service switch group. The LAG width of 64, distributed across a switch group of 16 Slingshot switches, was determined by the need for both L1 and L2 resiliency - each Rosetta switch in the Service switch group would connect to each Edge Router via a single cable carrying 2 links, thus ensuring a single cable or single link could fail without L1 isolation. Each cable connects to the routers via a single 400g physical port, carrying 2x200G logical links. Special keying was required to enable this functionality on the Arista.

The Edge Router pair itself is configured in an MLAG setup with VARP implemented. The single MAC address is then added to the Fabric Manager. VARP provides better traffic balancing and faster redundancy convergence, implementing active-active First Hop Router Redundancy to provide active/active unicast IP routing.

The 'first' Slingshot interface (hsn0) for each endpoint on the fabric is configured with a publicly routable IPv4/IPv6 address on both compute and login nodes. NIC and OS settings are adjusted to support the high bandwidth speeds the NIC is capable of with standard TCP, e.g. interrupt affinity, send/receive socket buffer size.

For connectivity northbound to the NERSC data center, we currently have 4x400G ECMP links to the data center routers from the Edge Routers for direct data streaming. This implementation delivers an aggregate bandwidth of 1.6Tb/s to and from the system. Site DNS and perimeter security are configured to allow SSH logins via the HSN border addresses of the login nodes. Login sessions are load balanced between the 40 login nodes using the methods described earlier in the section on Storage Networking.

VII. Planned work

While each GPFS server in NERSC's Community filesystem has up to four Infiniband HCAs and each of the gateways has two, only one HCA on each end is actually used for communication at any given time. This is due to a limitation in the IPoIB implementation of NIC bonding that only allows an active-backup configuration of interfaces. Solutions are being explored for ways to utilize all available links including managing interfaces individually instead of as part of a bond. Another area of exploration is GPFS's MCOT or Multiple Connections over TCP which allows multiple sockets to be opened between a client-server pair. One of the hurdles that need to be overcome is the explosion in the number of open sockets on servers from all ~5000 nodes of Perlmutter phase-2.

With 24 gateways routing traffic for ~5000 nodes in Perlmutter phase-2, network latency under heavy traffic load is a concern. If latency grows too large, it can cause GPFS timeouts that can lead to computes getting expelled from the GPFS cluster, which can lead to wider filesystem issue if a large number of computes were to expel simultaneously. Network queuing disciplines that are optimized to limit maximum latency such as CODEL could offer a potential solution.

NERSC's GPFS infrastructure is planned to migrate to Ethernet in the near future. This includes an additional 8x400G ECMP link between the HSN edge routers and the NERSC site network. There will be additional tuning required to optimize for this new network, including exploring the user of (soft)RoCE.

Realtime processing of data from remote experimental facilities is a major upcoming use case for Perlmutter. Design work is in progress to enable dynamic network routes that will allow these facilities to directly stream data to compute nodes.

VIII. Summary

HPE Cray EX systems have a flexible networking architecture highly suited to the evolving needs of HPC. This flexibility comes with increased complexity, which we manage and harness to our advantage using the methods and techniques described in this paper. By privatizing the CAN, we were able to set up a centralized management bastion server hosted in a resilient virtual machine infrastructure where administrators can login as themselves and perform administrative actions such as managing the Kubernetes cluster. The storage gateway and login load-balancing solution frees us from having to depend on dedicated hardware load-balancers while allowing us to expand and spread load even more by simply adding servers. By using IPVS, we can dynamically take login nodes out of service with minimal impact to users. Slingshot HSN integration into our data center allows users of the Perlmutter supercomputer to have seamless high speed access to the rich set of computing services that are provided at NERSC while also allowing them to create workflows with off-site data sources.