Enabling a SuperFacility with Software Defined Networking

Richard Shane Canon, Tina Declerck, Brent Draney, Jason Lee, David Paul, David Skinner
NERSC, Lawrence Berkeley National Laboratory
Berkeley, USA
Email: scanon@lbl.gov

***Abstract*—Experimental and Observational facilities are increasingly turning to high-performance computing centers to meet their growing analysis requirements. The combination of experimental facilities with HPC Centers has been termed the Super Facility. This vision requires a new level of connectivity and bandwidth between these remote instruments and the HPC systems. NERSC in collaboration with Cray has been exploring a new model of networking that builds on the principles of Software Defined Networking. We envision an architecture that allows the wide-area network to extend into the Cray system and enables external facilities to stream data directly to compute resources inside the system at over a 100 Gbs in the near-future and eventually reach beyond a 1 Tbs. In this paper will expand on this vision, describe some of the motivating use cases in more detail, layout our proposed architecture and implementation, describe our progress to date, and outline future plans.**

***Keywords*-SDN; HPC systems**

I. Introduction

Experimental and Observational facilities are increasingly turning to high-performance computing centers to meet their growing analysis requirements. Typically data has been transferred to dedicated end-points like data transfer nodes and stored in a campaign or scratch storage space. Once stored, the data can then be read from this storage and analyzed on the supercomputing resources. However, as the data rates from instruments continue to increase and the need for real-time response grows, these multi-hop approaches are no longer sufficient. In addition, projects associated with large collider facilities and other large distributed collaborations often use a distributed collection of resources for both data storage and analysis. Often data must transferred to the analysis resource at run-time. In the past, supercomputers like Cori would have had limited external access, but these new use cases demand much greater external access. NERSC in collaboration with Cray has been exploring a new model of networking that builds on the principles of Software Defined Networking. We envision a model that allows the external network to extend into the Cray system and enables external facilities to stream data directly to compute resources inside the system at over a 100 Gbs in the near-future and eventually reach a 1 Tbs. In this paper will expand on this vision, describe some of the motivating use cases in more detail, layout our proposed architecture and implementation, describe our progress to date, and outline future plans.

II. Background

Over the past decade, NERSC has formed close partnerships with a growing number of user facilities that are looking to NERSC to satisfy their growing demands for computing. This is often motivated by rapid increases in data generation rates from new instruments such as sequencers at the Joint Genome Institute or cameras at X-ray light source facilities like the Advanced Light Source or LCLS. These advances are often driven by Moore's Law improvements in the detectors. Some of these instruments have the ability to generate hundreds of megabytes of data per second today and promise to grow to gigabytes per second in the next five to ten years. Scientists at the ALS and LCLS want to leverage supercomputer class resources at NERSC to enable faster insight while the experiment is running. This enables the researcher to make adjustments in their experiment while they are still at the beam-line rather than discovering weeks later after they complete their analysis. In addition, researchers are looking at ways to couple observation and simulation together in real-time to improve the scientific process. This requires an unprecedented level of coupling between facilities. Within the Department of Energy Office of Science, this concept has been coined the Super Facility. In order to achieve this larger vision, NERSC needs to enable data to stream deep into the supercomputing system at parity with the generation rates by the instruments. Existing approaches have not been able to meet these demanding requirements. This has motivated NERSC to explore new models. Software Defined Networking is emerged in the larger networking space as a way to allow networks to more quickly adjust and adapt to user requirements.

NERSC is taking an incremental approach towards achieving the full vision of software defined networking that extends to the Cray systems. The early phases have focused on replacing the standard Realm-specific IP (RSIP) model used on the Crays with a new architecture that uses software-based routers running external to the system. Replacing RSIP was based on several factors including poor performance, minimal use by a broad community, and the lack of hooks to enable software-based control. The new software-based routers are running the Vyatta operating system (VyOS) which already supports many of the underlying capabilities required for SDN, but initially they are configured to do basic routing and network address translation (NAT). With these changes, NERSC has already demonstrated the ability to route packets from a single stream of a single compute node at over 25 Gbs. The service nodes that would have previously served as RSIP servers are now acting as bridge nodes that route packets from the internal Cray Aries network to an external ethernet network that connects to the Software-based routers. The routers have initially been configured to do standard masqueraded NAT. This allows compute nodes to talk efficiently initiate connections to external sources but doesn't permit inbound connections.

In the next, phase we are exploring ways to manage these inbound connections. The goal for this phase is to enable users to easily specify requirements for external addresses as part of their job script and have the system dynamically assign and associate addresses with the jobs. Careful attention should be paid to the crafting of this interface's usability. User job scripts (especially for workflows) are often themselves already complex in resource assignments. We propose a user-facing API which seeks simplicity. This capability should enable users to stream data externally from the system to compute resources. Our initial strategy is to leverage the APIs provided by the VyOS to dynamically configure the routers to map external addresses to the internal compute node address. The router would still be performing address translation, but would be doing a one-to-one map versus masquerading. Dynamic configuration will first be handled by a simple custom RESTful-based service that can easily be used by the batch system. This service will broker communications with the Vyatta routers and track and manage the pool of external addresses. Based on our progress, we hope to present results of this approach and describe the extended architecture in more detail.

III. Implementation

A. Approach

To enable these new models of coupling remote facilities requires a new approach to how networking is integrated into the HPC systems. The Cray systems have relied on Realm-Specific IP (RSIP) networking. This models has been shown to underperform in many cases, encounter issues when many compute nodes concurrently attempt to access external resources and lacks the programmatic flexibility that is needed to fully enable these new use cases. To addresses these limitations we have replaced RSIP with a new architecture that utilizes software-based routers and repurposes the RSIP servers into "bridge" nodes. We will briefly describe this architecture in more detail.

Software-based Routers

For Cori we have deployed ten software-based routers. Each of these has a single socket Haswell processor (8 cores, 3.5 GHz) and two dual-port 40Gb Intel NICs (check). These routers run the open-source VyOS software [1]. A single socket node configuration was chosen because the routing operations are bus and memory bandwidth constrained and multiple processors can lead to additional latency to transfer data between NUMA domains. The Intel NICs were selected because the Vyatta software can leverage the Data Plane Development Kit available for certain Intel NICs [2]. However, the routers are currently running the open source version of this software (VyOS) since the commercial version lacks support for the Intel 40Gb NICs.

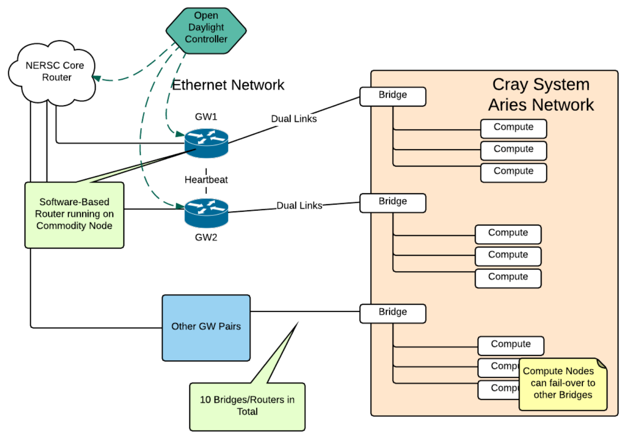

Fig. 1. Routing configuration used on the Cori system that utilizes software-based routers and bridge nodes connected to the high-speed Aries network.

Bridge Nodes:

The bridge nodes handle routing packets from the internal Cray Aries network to the external Ethernet network attached to the routers. These are effectively repurposed RSIP service nodes. Each is configured with a single Haswell Process (8 cores, 2.6 GHz) and two dual-port 40Gb Mellanox NICs.

B. Detailed Implementation

Changes are required in the compute nodes and bridge nodes to properly route traffic to the external routers. We will summarize those here.

Router Node:

The Router requires the following configuration settings:

- Configure the internal facing interface to reside in the same subnet (broadcast domain) as the high-speed network.
- Configure to do outbound Network Address Translation (e.g. Masquerading) for packets originating from the internal network.

Bridge Node:

The bridge nodes require the following configuration changes:

- External interfaces configured with a address in the same subnet used for the HSN but with a the netmask restricted netmask that includes that ideally just includes the router but not compute nodes.
- Enable Proxy ARP on the external interface
- Enable IP forwarding
- Set the default route to point to the associated external router.

Compute Nodes

The compute nodes require the following configuration changes:

- Add a static ARP entry that maps the IP address of the internal NIC on the router to the MAC address of the Aries interface on the bridge node. If there are multiple bridges and routers, these can be configured for all of the bridges/router pairs.
- Set the default route to IP address of the internal interface on the associated router for the compute node. If multiple routers are available, they should be distributed evenly across all routers.

C. Example Configuration

To help illustrate these configuration changes, let's assume a scenario where the system has a single bridge and router pair (see Fig. 2). The HSN subnet is 10.128.0.0/23. The router is configured with 10.128.255.1/23 on the internal interface (eth0). The bridge node HSN MAC address is 00:01:01:00:00:0d and the IP address of the external interface (eth0) is configured as 10.128.1.2/24. The key parts of the router config can be seen in Fig. 3. Likewise the configurations for the bridge and compute node can be seen in Fig. 4 and 5, respectively.

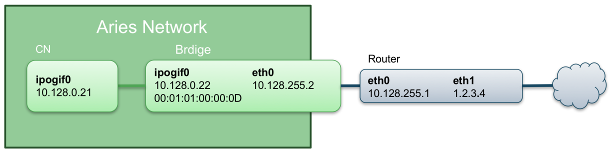

Fig. 2. Illustration of the networking configuration using a “bridge” node to provide connectivity to an external router.

D. Enabling In-bound Routing

To achieve the full vision of the SDN project, inbound routing must be supported. Furthermore, it should be possible to dynamically assign these addresses and integrate this control with the HPC resource manager (SLURM). To provide this control, we have implemented a simple RESTful interface that can allocate and assign external addresses to a compute node and later release them. This service in still in a prototype state. We will briefly its design and implementation.

To enable this service, we wanted a simple interface that ultimately could be integrated with SLURM. This service has been implemented in Python using the Flask framework [3]. The service uses a Mongo Database to track the state of the external addresses in order to avoid conflicts. The service uses Munge to authenticate requests and protect against unauthorized access [4]. Munge is already used for authentication by SLURM, so we leverage the existing deployment. Unfortunately, the open-source VyOS software does not offer a programmatic interface. There, the SDN gateway interacts with the routers via SSH using expect-style methods to configure the routers.

Currently, the service supports a limited set of commands. A summary of these is shown in Table I. When a “associate” request is made to the service, this results in the following actions:

```
interfaces {
    ethernet eth0 {
        address 10.128.255.1/14
        description "Int interface"
    }
    ethernet eth1 {
        address 1.2.3.4/24
        description "ext interface"
    }
}
nat {
    source {
        rule 100 {
            outbound-interface eth1
            protocol all
            source {
                address 10.128.0.0/14
            }
            translation {
                address 1.2.3.4
            }
        }
    }
}
```

Fig. 3. Key parts of the VyOS router configuration

```
ifconfig eth0 10.128.255.2 netmask \
   255.255.255.0 up
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 1 > \
  /proc/sys/net/ipv4/conf/eth0/proxy_arp
route add default gw 10.128.255.1
```

Fig. 4. Configuring the bridge.

1) Authenticate the request via Munge
2) Find an available address or return an error if none are available
3) Lookup the router associated with the requesting node
4) Configure the address on the external interface of the associated router via ssh.
5) Configure an in-bound NAT rule for the external address to the internal address.
6) Configure an out-bound NAT rule from the internal address to the external address.
7) Commit the operations and return the external address in the response.

A release request essentially does the same set of operations

```
/sbin/arp -s 10.128.255.1
00:01:01:00:00:0D
/sbin/route add default gw 10.128.255.1
```

Fig. 5. Configuring the compute node.

in reverse and returns a ”released” status message upon success.

In the future, we will integrate this service with SLURM. We envision the user being able to pass options to their batch job request to request an external IP. Once the job has started, it will be able to access the allocated address via an environment variable. The application could then publish this address to a coordination service or a dynamic DNS service.

TABLE I
GATEWAY API (ALL METHODS ARE CURRENTLY IMPLEMENTED AS GET REQUEST)

Command	Purpose
/associate/	Allocate and associate an address to the requesting compute node
/release/	Release and free
/status/	Return the status for public addresses including current mappings

IV. PERFORMANCE

A major goal of the SDN project was to improve performance over the existing RSIP-based approach and to enable future use cases requiring high ingest bandwidths. We have made baseline measurements of the previous RSIP configuration and compared it with this new approach. In some case, we have also measured the performance on a directly connected login node to determine the potential performance for a host that is directly connected to the wide area network. In general, the new approach provides superior performance and approaches the performance of the directly connected node configuration. However, there are still some use cases exhibiting poor performance compared with the directly connected node. It is worth noting that the new approach still outperforms the previous RSIP approach in all cases we have measured.

A. IPERF

Iperf is a low-level IP network performance measurement tool [?]. It is typically used to determine the baseline network performance between two end points. In these tests we are using iperf version 3 with no additional command-line options. Table II summarizes the results.

TABLE II
IPERF MEASURED PERFORMANCE.

Configuration	Performance (Gbps)
RSIP	4.5
Directly Connected	34.7
SDN Configuration	25.8

B. Grid FTP

GridFTP is part of the Globus Grid Toolkit. For this test, we measured transfers from an ESnet test endpoint that supported 9000 Byte MTU (e.g. Jumbo Frames). Similar to the iperf benchmark, we report the performance for RSIP, a directly connected login node, and the new approach. We test against two end points. One end-point is hosted at Berkeley lab and thus has very low latency from NERSC. The other is hosted at CERN, and has much higher latency. The bandwidth delay product dictates that a high latency path will achieve lower bandwidth if when using a common window size. So, not surprisingly, the CERN results are much lower. The results are summarized in Table III.

TABLE III
PERFORMANCE DOWNLOADING A FILE VIA GRIDFTP.

	Performance (MB/s)	
Configuration	LBNL	CERN
RSIP	540	13.0
Directly Connected	750	19.5
SDN Configuration	550	13.0

C. TCP Backlog Drops and Tuning

As can be seen from the above results, the new approach improves performance significantly for many use cases. However, some use cases continue to perform poorly. For example, downloading a file from a web server connected at a standard frame size (1500 Byte MTU) will perform achieve around 1/10 of the bandwidth compared to the same test performed on a login node directly connected to the external network. An example of this is shown in Table IV. Here a large file is downloaded from an end-point located hosted at CERN. Unlike the benchmark performed with GridFTP, this web server is running at a standard frame size. The reported numbers are observed sustained performance after TCP ramp up. The performance for the RSIP and SDN results are extremely noisy, implying dropped packets or similar issue.

TABLE IV
PERFORMANCE WITH AND WITHOUT ADDITIONAL TUNING.

Configuration	Performance (MB/s)
RSIP	0.2
Directly Connected	20
SDN Configuration (Default)	1
SDN Configuration (Optimized)	25

To diagnose this issue, TCP traces were collected. The results of one trace is shown in 6. As expected, retransmits are occurring. This explains the erratic performance and poor performance. On closer inspection, the drops are discovered to occur after arriving at the compute node. The trace shows that the packet safely arrives all the way to the compute node, but then a retransmit is still observed. Further investigation showed that this was due to the packet being dropped due to insufficient TCP buffer space. This can be seen in the “TCP Backlog drop” counter ($netstat - s$). This typically occurs because the application is not reading packets off of the socket quickly enough.

This issue was reported to Cray and engineers explained that the driver that provides the IP interface for the Aries interconnect (e.g. ipogif) uses a fixed buffer size for all packets set to the MTU size. Since the interface is typically configured

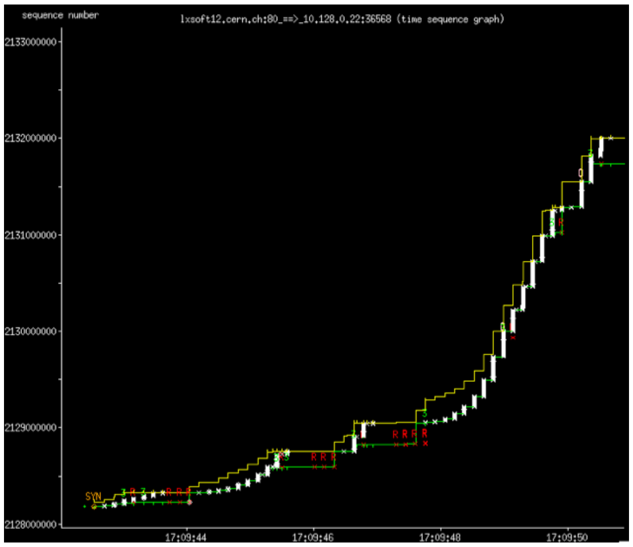

Fig. 6. Graph of TCP trace illustrating retransmits (red Rs) due to dropped packets.

at 64k, the default maximum per-connection memory limit can be exhausted by a few hundred packets. As an experiment, the maximum per-connection was increased to 256MB (16x larger than the default) on Cori's test and development system. This was done by increasing the last number in the tcp_rmem and tcp_wmem in $/proc/sys/net/ipv4/$. Applying these settings on the test system alleviated the problem. The performance increased by over 10x and even out performed the directly connected login node. The results can be seen in the final row of Table IV. Based on these results, the same changes were applied to the much larger Cori system. Unfortunately, Cori did not achieve the same performance gains. Backlog drops are still occurring and the performance is roughly the same as without the setting. NERSC staff are continuing to work on this issue.

V. Discussion

The early results of this work are mixed. Many of the use cases show signifcant improvements, but work remains to diagnose and address the remaining performance issues. The prototype API service that enables addresses to be dynamically allocated and associated with compute nodes provides a new level of flexibility and control. However, additional work is needed to harden this service, integrate it with the SLURM resource manager, and move it into production.

In future phases, we will explore using more standard Software-Defined Networking concepts and software. Our architecture includes an OpenDaylight (ODL) Controller which provides a platform for building end-to-end SDN applications. The ODL Controller is not used in the early phases of the project since it introduces additional complexity. There are many open questions about how exactly to integrate many of these components with Cray systems. In the larger paper and presentation, we will describe basic SDN concepts, potential ways of integrating these with the Cray, and our future plans including thoughts on extended this architecture to multiple sites to enable the Super Facility vision.

VI. Conclusion

Their is an increasing need to integrate the capabilities of experimental facilities and high-performance computing centers to keep pace with growing computational requirements, enable more rapid discover, and enable coupling of simulations and experimental data. A key barrier to enabling this new model is lowering the barriers between the facilities and ensuring that data can flow at extremely high-data rates between locations. We have designed and deployed a new architecture that will enable this vision. While there is still much work left to be done, the preliminary progress already demonstrates that new approach can achieve high-performance and flexibility. Building on this model, we hope to provide experimental scientists an ability to seamless integrate these resources and enable HPC resources like Cori to effectively become an extension of the experimental apparatus. Achieving tis vision will open doors to new scientific discovery.

Acknowledgment

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

[1] "Vyos open source router," https://vyos.io/.
[2] "Data plane developer kit," http://dpdk.org/.
[3] "Flask web development framework," http://flask.pocoo.org/.
[4] "Munge uid 'n' gid emporium," https://dun.github.io/munge/.