Enabling a SuperFacility with Software Defined Networking

Shane Canon

Tina Declerck, Brent Draney, Jason Lee, David Paul, David Skinner

CUG 2017
SuperFacility - Defined

Combining the capabilities of multiple user facilities in a seamless manner to accelerate and enable new models of scientific discovery and innovations.
Examples

Coupling experimental and observational facilities with advanced networking and high-performance computing and analytics capabilities.
Technical Barriers

- User Management
- Data Transfer
- Data and Metadata Management
- Porting applications and workflows
- Workflow Execution
Networking Use Cases

Supporting SuperFacility and data-intensive use cases requires new modes of network access compared to traditional HPC modeling and simulation.

- Jobs from ATLAS, ALICE, STAR and others must communicate with remote data servers and job managers.
- In the future, LCLS2 will need to stream data at ~ 1 GB/s direct to memory or burst buffer for real-time processing.
- In the future, network must be provisioned like compute/memory resources to allocate and control access and integrate with ESnet.
Objective

- Compute nodes can access external services and ingest data at high-bandwidths and high connection rates.

- Compute nodes can also be accessed by external systems (e.g. for streaming uses cases).

- Bandwidth and access to compute nodes can be engineered based on job placement and user needs.
Evolution of Data Transport

Instrument Side Storage → DTN → DTN → NERSC “Project” FS

Instrument Side Storage → Burst Buffer
Improving External Network Access on Cori
Problems with current RSIP model

• Poor performance in some cases*
• Port exhaustion for many short connections*
• Lacks Fail-over support
• Lacks Flexibility and Programmability

* Has improved over past year
Approach

• Deploy Software-based Routers
• Repurpose RSIP-type nodes to act as "bridges" between HSN and external routers
• Develop API service to enable resource manager (SLURM) to manage router configuration
• Extend architecture to eventually couple with software-defined network enabled infrastructure to the border and out to ESnet
What was required

Bridge
• Enable Proxy ARP
• Configure routes
• Enable IP forwarding

Compute Nodes
• Add ARP entries for Bridge/Gateway pairs
• Change default route to gateway

Router
• Deploy and configure software-based routers
Key Configuration Information

Compute Nodes Settings

/sbin/arp -s 10.128.255.1 \
00:01:01:00:00:0D

route add default gw 10.128.255.1

Bridge Node Settings

ifconfig eth0 10.128.255.2 \
etmask 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
Status and Performance

• Outbound NAT is in production on Cori
• Performance is typically ~5x better than RSIP*
• Some tests and use cases showing great performance
  – Single stream Iperf - 25 Gbps (CN <-> Login)
  – Local Globus Transfers – 550 MB/s (single stream), 1 GB/s (multiple streams)
• Some tests and use cases show poor performance (more later)
Towards SDN
Adding More Software

- Still need a mechanism to allow control of routes via resource manager
- Approach: Develop simple REST-like API service to configure router
Prototype “SDN” API Service

“SDN” Gateway
• REST-like API interface
• Tracks available and used external addresses
• Python Flask
• Munge Authentication (HTTP Header)
• Credentials for VYOS Router
• Issues Expect-based commands via SSH to Router
API

• Auth Header encrypted by munge which includes user information and IP address

End Points

• /associate/ - Allocate an address and map (1-to-1 NAT) to the compute node IP
• /release/ - Release the IP address associated with the compute node
• /status/ - Show current mappings
• /addresses/ - Show unallocated addresses
• Extend integration to interact with SDN controller (e.g. OpenDaylight, Ryu). This could include enabling OpenFlow-based protocols to enable a fast path through internal networks at both ends and across ESNet.
**SDN Definition**

- A software-defined networking (SDN) architecture (or SDN architecture) defines how a networking and computing system can be built using a combination of open, software-based technologies and commodity networking hardware that separate the control plane and the data layer of the networking stack. (from SDx Central)

- Typically SDN uses open standards such as OpenFlow to communicate and manage data flows.
Performance Issues
Performance Challenges

• Observation: performance for many operations are still slow, especially when talking to remote endpoints with standard MTU sizes (1500 bytes).

• Example: `wget/curl` against a CERN URL is 5x slower compared to the login node performance.

• Data: Poor performance is correlated with TCP backlog drops on the compute node (netstat –s).

```
canon@mom1:~> netstat -s|grep Backlog
TCPBacklogDrop: 7
```
Performance

- TCP traces show that packet arrives but then has to be retransmitted
Diagnosis and Improvement*

• With Cray input, concluded the TCP buffers were being exhausted.
• Ipogif interface uses ~64k MTU. The upper limit for the TCP buffers per connection is ~16M
• Increasing this to 256MB improved performance by 5-10x (on the TDS system)
• WIP: Improvements didn’t translate to Cori. Still see roughly ~10x slow down.
Next Steps

Near Term
• Diagnose and fix performance issue
• Deploy configuration on Edison

Mid Term
• Testing SDN Controllers and Integration
• Exploring Slurm Integration

Long Term
• Extending to LAN and WAN
A NERSC Cray data system is transparently accessible to any scientist in the world, as though it was on their own network.

- To do this we need to have a fully customizable routing into the Cray that can be used as part of a dynamic circuit between a remote scientist, instrument or data source and the internal Aries network.

- We need the ability to control the routing layer through a combination of the batch system and software defined networking (SDN) in order to engineer traffic from a remote site to a scheduled job on a Cray supercomputer.
Questions?