Dynamic Network Perimeterization

Isolating Tenant Workloads with VLANs, VNIs, & ACLs

Dennis Walker
HPC Solutions Engineering
Hewlett Packard Enterprise
Las Vegas, NV - United States
dennis.walker@hpe.com

Stephen Han
HPC Slingshot
Hewlett Packard Enterprise
San Jose, CA - United States
seung-bong.han@hpe.com

Siri Vias Khalsa
HPC Solutions Engineering
Hewlett Packard Enterprise
London, England - United Kingdom
sirivias.khalsa@hpe.com

Vinay N Karanth
HPC Slingshot
Hewlett Packard Enterprise
Bangalore, Karnatak - India
karanth@hpe.com

Nikhil Mukundan
HPC Slingshot
Hewlett Packard Enterprise
Nashua, NH - United States
nikhil.mukundan@hpe.com

Atif Ali
HPC System Management
Hewlett Packard Enterprise
Washington D.C. - United States
atif.ali@hpe.com

Amit Jain
HPC Slingshot
Hewlett Packard Enterprise
Bangalore, Karnataka - India
amit.jain2@hpe.com

Vishal Bhatia
HPC Slingshot
Hewlett Packard Enterprise
Charlotte, NC - United States
vishal.bhatia@hpe.com

ABSTRACT

Modern high-performance computing (HPC) environments increasingly require dynamic, secure multi-tenancy to support mixed-sensitivity workloads across government, academic, and private-sector institutions. This paper presents an architecture that leverages technologies such as Slingshot, HPCM, CSM, and ClusterStor to enable flexible partitioning of compute, storage, and network resources. Techniques including dynamic Virtual Network Identifier (VNI) allocation, VLAN enforcement, IPtables-driven access controls, and centralized configuration management are explored. This paper demonstrates how distributed management, automation, and defense-in-depth strategies can meet stringent security requirements while maintaining high availability, scalability, and operational efficiency.

INTRODUCTION

Multi-tenancy is increasingly essential for public-sector organizations, governments, educational institutions, and some private-sector companies. In such cases, hosting organizations require robust security to isolate activity between tenants or for critical workloads to execute concurrently with lower security clearance activity. New features in Slingshot, CSM, and other products enable dynamic tenant partition resizing of the infrastructure supporting workloads, including compute node groups, data-at-rest (storage), and data-in-motion within high-speed and management networks.

The advent of flexible node provisioning per tenant also calls for dynamic network perimeter adjustments. This can be achieved by effectively partitioning high-speed networks (HSN) and management networks through automation and APIs. Readjustments to compute node partitions can be applied quickly based on demand, which results in more efficient resource utilization and improved security.

In such environments, sensitive transactions are further insulated from lower-clearance activities by isolating storage, management, and HSN (Slingshot) network traffic. This creates secure zones for specific security and operational needs and optimizes data flow efficiency. It mitigates data breach risks and enhances compliance with security regulations while keeping partitions flexible to ever-

changing demand. It generally raises the security posture, even in single-tenant environments.

This paper details how product-agnostic, version-controlled configuration data can be used to dynamically isolate infrastructure resources supporting workloads, including compute node groups, data-at-rest (storage), and data-in-motion within high-speed and management networks. Complete dynamic network segmentation will be applied at various levels of infrastructure from chassis, nodes, and within the OS.

We will also examine the lifecycle of node reprovisioning, moving it from one tenant to another and applying a unique policy. We'll compare and contrast the improved security risks and configuration flexibility when configuring boundaries at the switch network and node levels.

Finally, the paper reviews an architecture consisting of Slingshot, HPCM/CSM, ClusterStor, and other products that showcases how dynamic segmentation is enabled in such solutions.

CCS CONCEPTS

- Firewalls
- Network Security
- Network Protocols
- Network Architectures
- Network Manageability
- Network Management
- Link-layer Protocols
- Network Layer Protocols
- Transport Protocols
- Cross-layer Protocols

KEYWORDS

Multi-tenancy, Security, Networking, Partitioning, Data-in-motion, Node-groups, Orchestration, Automation, Isolation, Slingshot, Aruba, HPCM, CSM, ClusterStor, GPFS

1 USE CASES FOR NETWORK PERIMETERIZATION

1.1 Isolating Tenant

This model is also referred to as a "multi-tenant" or "Coke and Pepsi" model, in which two or more potentially competitive organizations are sharing the same system. In this case, the desire is to break the system into isolated domains which cannot communicate directly with each other over the HPC fabric and management networks.

1.2 Isolating Administrative Functionality

Isolating users of a high-performance computing (HPC) system from administrative functionality is critical for maintaining the integrity, stability, and security of the environment. HPC systems often support hundreds or thousands of concurrent users running complex workloads that can strain resources and reveal vulnerabilities. Allowing users access to administrative controls—such as system-level configuration, hardware management, workload management, or even local root permissions—introduces significant risk. An inadvertent command, misconfiguration, or malicious action could disrupt active workloads, corrupt shared file systems, or compromise sensitive data and credentials. By strictly segmenting user privileges, administrators can ensure the core system remains protected from unintentional or unauthorized changes.

1.3 Isolating Services

Network access and transmission to global HPC services like DAOS, DVS, and Lustre file systems can be further isolated through the high-speed network and down to the requested job. This is facilitated through the use of "published" VNIs, used to communicate with global services via HPC traffic, and are publicly available to all applications launched by a work-load manager. The hard VNI enforcement for service VNIs is done during the initial setup and not done at a per application or job level initialization. This enforcement makes sure that the service is accessible to the applications launched by this workload manager only and not to any other application outside its domain.

1.4 Isolating Job Steps

A single application instance launched by a workload manager job script, to run across one or more compute nodes is referred to here as a job-step. The job script can launch multiple job-steps. Each such job-step or multi-node application will need an "application" VNI to allow the processes of the application (running on different nodes) to communicate with each other for data-sharing and synchronization.

1.5 Isolating Job Groups

A workload manager "job" can contain multiple concurrently running job-steps or applications. Some of these may need to communicate with each other, so every multi-application job will need a "common" VNI shared by all applications within that job. A typical example is two different applications communicating with each other using MPI_comm_connect/accept.

2 ARCHITECTURAL FEATURES FOR NETWORK PARTITIONING

2.1 Slingshot Partitioning Features

Slingshot implements a logical partitioning scheme meaning that the physical resources of the network, such as the switch ports, are shared among the partitions. The partitions can be created or destroyed dynamically based on the requirements of the system.

This logical partitioning scheme identifies the traffic belonging to a particular partition and prevents the traffic from being delivered to any other partition through an encapsulation header carrying a partition identifier.

For HPC traffic, a unique 16-bit identifier called Virtual Network Identifier (VNI) is used in a Slingshot L2 header. Whereas, for Ethernet traffic such as ROCEv2, it is the standard VLAN tag described in IEEE 802.1q specification.

2.1.1 Slingshot VNI

In a switch level VNI enforcement, a VNI filter can be set on either the ingress edge port or the egress edge port of the Slingshot fabric. VNI enforcement corresponding to a tenant involves setting the VNI filter on every edge port which connects to a node belonging to the tenant.

Fabric manager of a Slingshot network exposes following constructs for flexible configuration of VNI filtering. This scheme has been introduced to cater to different use case requirements to partition the network.

VNI partition

VNI partition is a logical partition that can be created either by a network operator plugin or any other management service with admin privileges to reserve a set of VNIs either by count or range.

In case of a multi-tenant environment, a VNI range can be reserved at tenant level. In another use case, a similar range can be reserved by a management service which can further allocate per job level VNIs to implement network partitioning at application level. VNI partition can have one or more VNI blocks.

VNI block

VNI block is a subset of VNI partition which forms a VNI filter programmed in Slingshot switch hardware. A VNI block represents actual switch level VNI enforcement.

Image 2.1.1 - Diagram depicting how VNI partitions help isolate tenants.

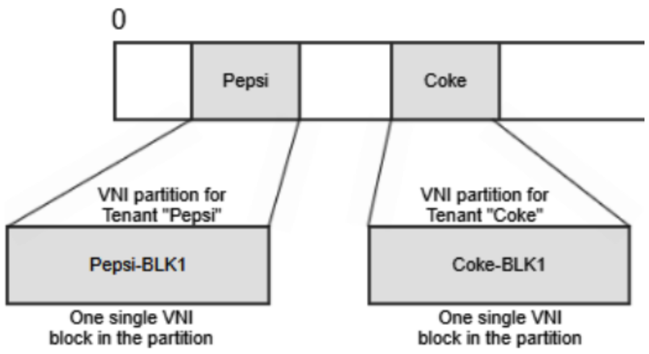
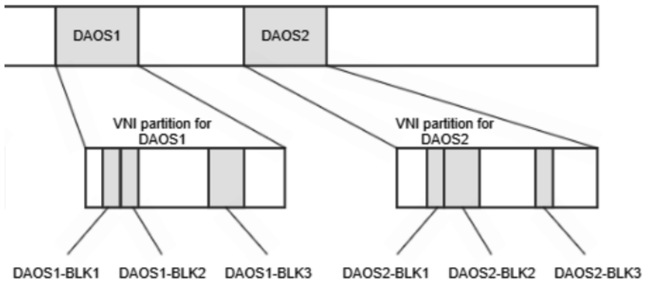

0
Pepsi
Coke
VNI partition for Tenant "Pepsi"
VNI partition for Tenant "Coke"
Pepsi-BLK1
Coke-BLK1
One single VNI block in the partition
One single VNI block in the partition

Image 2.1.2 - Diagram depicting how VNI blocks isolate access to services



In one use case, there are 2 VNI partitions created corresponding to tenants called Pepsi and Coke. There is one single VNI block in the partition with full range of VNIs which means all VNIs are enforced at switch level to provide the isolation.

In second use case, the VNI partition is broken into multiple blocks by a management service. These blocks can be added or removed dynamically multiple times during the lifecycle of the service to apply and remove switch level enforcement at a VNI block level.

Create a VNI
Example of creating vni partition and reserve 200 VNIs

Request:

```
POST /fabric/vni/partition

{

  "partitionName":                        "WLM1",
  "description": "VNIs managed by WLM1",

  "vniCount”: 200,

  "edgePorts": [0x84a000, 0x8ac000, 0x1163000,
0x1532000, 0x3278a000, 0x64294000]

}
```

Response:

```
{
  "partitionName" : "WLM1",

  "description" : "VNIs managed by WLM1",

  "vniRange" : ["2000-2199"],

  "vniCount" : 200

  "edgePorts":  [0x84a000,  0x8ac000,  0x1163000,
0x1532000, 0x3278a000, 0x64294000]

}
```

Example of enforcing VNIs on the port

Request:

```
POST /fabric/vni/block

{

  "blockName" : "WLM1-BLK1",

  "vniRange": ["2000-2100"],

  "partitionName" : "WLM1",

  "portDFAs"  :  [0x84a000,  0x8ac000,  0x1163000,
0x1532000, 0x3278a000, 0x64294000]

}
```

Response:

```
{

    "blockName" : "WLM1-BLK1",

    "partitionName" : "WLM1",

    "vniRange" :  [ 2000-2100 ],

    "portsDFAs"   :      [0x84a000,   0x8ac000,
0x1163000, 0x1532000, 0x3278a000, 0x64294000]

    "enforcementTaskServiceLink" : <String>

}
```

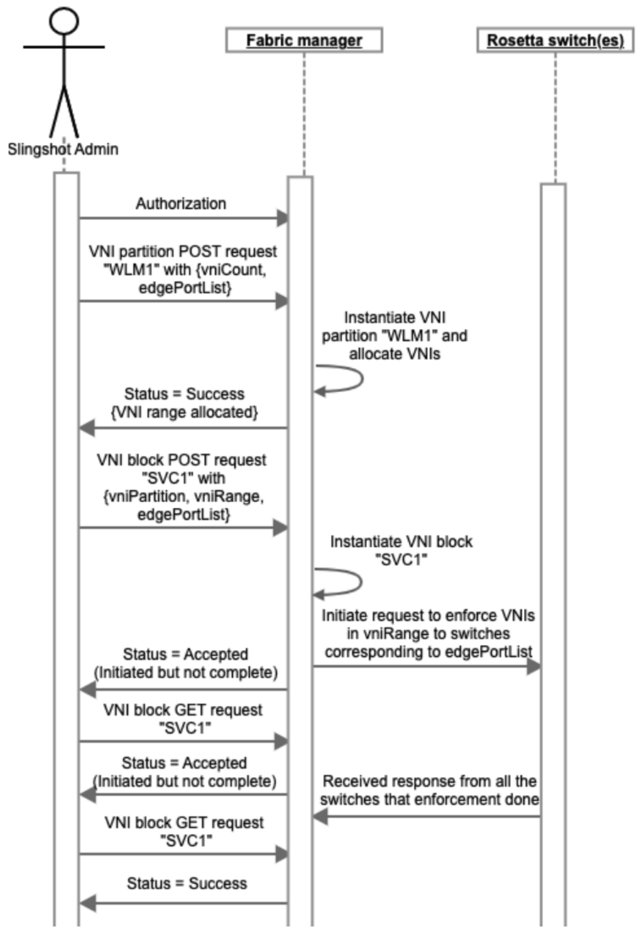

Slingshot Admin
Fabric manager
Rosetta switch(es)
Authorization
VNI partition POST request "WLM1" with {vniCount, edgePortList}
Instantiate VNI partition "WLM1" and allocate VNIs
Status = Success {VNI range allocated}
VNI block POST request "SVC1" with {vniPartition, vniRange, edgePortList}
Instantiate VNI block "SVC1"
Initiate request to enforce VNIs in vniRange to switches corresponding to edgePortList
Status = Accepted (Initiated but not complete)
VNI block GET request "SVC1"
Status = Accepted (Initiated but not complete)
Received response from all the switches that enforcement done
VNI block GET request "SVC1"
Status = Success

Image 2.1.3 - Process diagram depicting what is happening to the network infrastructure when Slingshot Fabric provisions VNIs.

2.1.2 Slingshot VLAN

The VLAN filtering scheme supported in Slingshot fabric follows standard VLAN tagging documented in IEEE 801.q specification. A 12-bit VLAN tag is used to enforce VLAN filtering at ingress and egress edge ports of the Slingshot fabric.

Fabric manager of a Slingshot network exposes vlan-policy as one of the port policies that can be used to configure the allowed VLANs on edge ports. For a tenant, the edge ports connected to the nodes belonging to the tenant can be assigned a vlan-policy to apply relevant allowed VLANs.

Create a Slingshot VLAN

VLANs are provisioned in the Slingshot switch fabric using the Fabric Manager controller. The process involves creating the VLANs within the fabric, which is the first step in enabling VLAN functionality.

Use the `` `fmctl create vlans` `` command.
The command requires you to specify:

- name: A name for the VLAN
- status: This should be set to ONLINE
- id: The VLAN ID

For example, to create VLANs with IDs 1, 2, and 5, use the following commands:

```
slingshot-fabric-manager# fmctl create vlans
name=RedNetwork status=ONLINE id=1

slingshot-fabric-manager# fmctl create vlans
name=GreenNetwork status=ONLINE id=2

slingshot-fabric-manager# fmctl create vlans
name=BlueNetwork status=ONLINE id=5
```

After executing the command, the output will show details about the created VLAN, including its documentSelfLink, id, name, and status

Creating the VLAN is part of a larger process to configure and enable VLANs in the fabric. The complete recommended steps to configure VLANs include:

- Create VLANs in the fabric using the fmctl create vlans command
- Create or modify a port policy to include the required VLAN settings (allowedVlans, isUntaggedAllowed, and nativeVlanId)
- These settings are saved aggregated inside a port policy
- Apply the port policy containing the VLAN settings to specific edge ports. VLAN settings are only valid on edge ports; any setting on a fabric port will be ignored.
- Enable VLAN in the whole fabric by setting the global vlanEnabled setting to true in the topology-policies/template-policy

When vlanEnabled is true, every port in the fabric will apply the VLAN settings defined in its port policy

Immediately after performing any VLAN configuration change in the Fabric Manager, the settings are pushed to every switch and come into force without needing to restart any element of the fabric.

VLANs can be configured at any time on a running system if you need multiple virtual networks, however, changing the VLAN configuration on a running system may cause traffic disruption. Alternatively, if the configuration is known in advance, it is recommended to set up VLANs when configuring port policies before bringing up the fabric.

2.1.3 Slingshot Network Operator

The Slingshot Network Operator provides automated VNI partitioning for tenants defined within system management. It has been recently released as an integration to CSM's multi-tenancy features and is detailed in the CSM Partitioning Features section below.

It demonstrates a design pattern that can be implemented, in part, with other system management platforms.

2.2 HPCM Partitioning Features

HPCM defines the management network which facilitates administrative tasks, node provisioning, configuration, monitoring, and command execution across the cluster. It connects the infrastructure nodes (admin and leader nodes) to the compute nodes and other managed devices like switches and controllers.

Although HPCM does not natively provide network partitioning capabilities, it can be extended using third-party tooling. This has been implemented in a number of organizations and is covered in the case study below.

2.4 CSM Partitioning Features

CSM contains a limited feature set for managing tenancy, which now includes network partitioning of high-speed networks and management network partitioning on the horizon.

2.4.1 Tenant Partition Management (TAPMS)

CSM's multi-tenancy feature set is built around the TAPMS, a Kubernetes-based operator which manages tenants linked to system resources defined by HSM partitions, HSM groups, and specific component xnames. Other resources, like HSN network partitioning, CFS, Vault, SLURM job schedulers, etc., are managed either through additional Kubernetes operators watching for changes in TAPMS or via TAPMS's tenant lifecycle callback hooks (1) to external services.

Provisioning a Tenant

Tenant Resource is a tenant definition. It has various fields which are required to create a tenant in a multi-tenant environment. Slingshot network operator is interested in tenant name and nodes assigned to tenant. Nodes are defined in the field xnames.

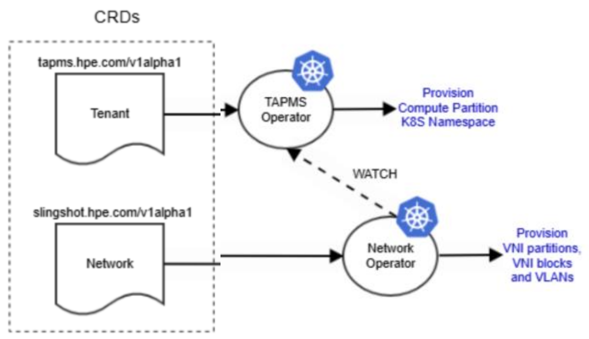
Example 2.4.1 - A tenant definition in yaml
```
apiVersion: tapms.hpe.com/v1alpha3
kind: Tenant
metadata:
name: vcluster-tyrten02
spec:
childnamespaces:
- slurm
- user
```

```
tenantname: ExampleTenant1
tenanthooks: []
tenantresources:
- enforceexclusivehsmgroups: true
   hsmgrouplabel: tyrten02
   type: compute
   xnames:
   - x9000c1s0b1n0
   - x9000c1s0b1n1
- enforceexclusivehsmgroups: true
   hsmgrouplabel: tyrten02
   type: application
   xnames:
     - x3000c0s29b0n0
```

2.4.2 Slingshot Network Operator

Slingshot network operator is a Kubernetes operator that continually watches the TAPMS operator for tenant addition and deletion events and based on that modifies the desired state for VNI blocks / partitions or VLANs in the Slingshot fabric. The event driven state machine discussed in this section can be implemented using any other technology. Slingshot network operator is a reference implementation that helps understand the desired configuration needed and the events to act on to set the Slingshot fabric for multi-tenancy.

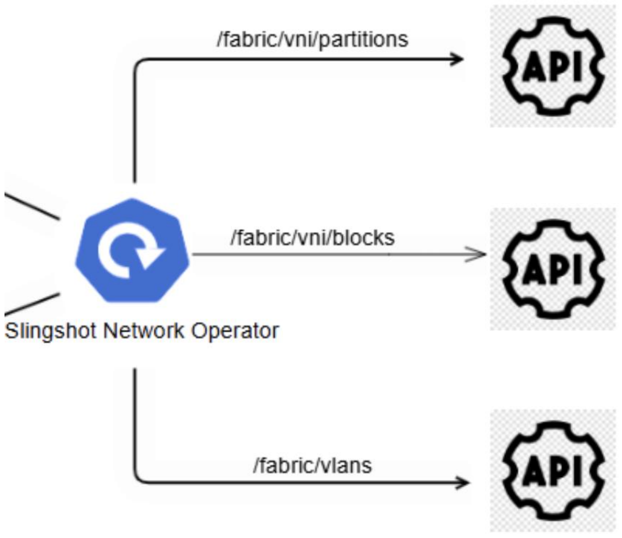
Image 2.4.2 - Diagram depicting the interaction between the Slingshot Network Operator and CSM's TAPMS tenancy operator.


CRDs
tapms.hpe.com/v1alpha1
Tenant
TAPMS Operator
Provision Compute Partition K8S Namespace
WATCH
slingshot.hpe.com/v1alpha1
Network
Network Operator
Provision VNI partitions, VNI blocks and VLANs

It is configured to observe 2 CRDs i.e. TAPMS and Slingshot CRD. It acts on following events:

- Creation or Deletion or Update of a tenant in TAPMS CRD
- Update of network partition configuration in Slingshot CRD

Image 2.4.3 - Operator reconciliation, it invokes Fabric manager REST APIs to configure VNI partition / blocks or VLANs as shown in the diagram below.



It then relays the status back to TAPMS and updates its internal state to show status.

2.4.3 Network Partitioning Security Model

For tenant vni-blocks and vni-enforcement-sync-tasks resources, the Brewer-Nash security access model shall be applied to remedy the conflict of interest among tenants. The model shall ensure each tenant resource shall be protected from each other.

The following describes the default roles and domains of control.

Infrastructure Administrator

The Infrastructure Administrator role has administrative control of following resources:

- Compute node
- Non compute node
- Storage node
- Management network
- HSN network

Slingshot Administrator

The Slingshot Administrator role has administrative control of following resources:

- HSN Network
- Slingshot Fabric Manager
- Slingshot switch
- Slingshot host software and HSN NIC
- Slingshot Fabric Manager resource

- VniPartition
- VniBlock
- VniEnforceSyncTask

Tenant Administrator

The Tenant Administrator actor is role delegated by Infrastructure Administrator to manage the resource partitions created by the Infrastructure Administrator for the Tenant User to consume. The resources managed by Tenant Administrator is limited to HSN resources defined in Fabric Manager.

- Slingshot Fabric Manager resource
- VniBlocktenant
- VniEnforceSyncTasktenant

Authentication and authorization

HSN resource is accessible through Fabric Manager Northbound REST APIs. Fabric Manager API gateway shall provide the confidentiality and integrity through Mutual TLS (mTLS) and OpenID Connect-based authentication and authorization for every resource request over REST API.

3 Case Study

Recently, Hewlett Packard Enterprise (HPE) successfully deployed a large-scale, multi-zone, distributed supercomputing system, encompassing more than 10,000 compute nodes. To meet the stringent requirements for high availability, optimized performance, and strict security controls, the system architecture was deliberately segmented across multiple management platforms — specifically, four separate instances of Cray System Management (CSM) and two instances of HPE Performance Cluster Management (HPCM).

This distributed management approach provided several critical advantages. It allowed for operational redundancy in case of localized failures, minimized performance bottlenecks by scaling management services, and introduced multiple layers of isolation to enhance overall system security.

Network partitioning was a key strategy employed to address two primary use cases:

- **Separation of User and Administrative Access:** To reduce the risk of privilege escalation and protect administrative operations, network segments were carefully designed to isolate user-facing workloads and portals from the administrative interfaces, management nodes, and underlying system services. This separation ensures that users cannot inadvertently or maliciously access system-level controls, maintaining a secure operational boundary between end-users and system administrators.
- **Segmentation Based on Security Clearance:** To support environments with varying levels of data sensitivity and user trust, the system further partitioned the network to enforce access control based on security clearance levels. Users with lower levels of clearance were restricted from seeing or interacting with compute nodes, jobs, datasets, or storage resources associated with higher-classification workloads. This strict separation minimizes the risk of unauthorized data exposure and aligns with compliance frameworks requiring logical and physical isolation of sensitive information.

Overall, the distributed management design and targeted use of network partitioning allowed HPE to deliver a resilient, scalable, and secure supercomputing environment capable of supporting both scientific research and classified workloads within a unified but compartmentalized system.

3.1 Topology & Partitions

The deployed supercomputing environment spans two physically distinct datacenters, referred to as "zones." Each zone is designed to operate autonomously and contains the core components necessary to support both high-performance computing workloads and administrative functions. Specifically, each zone includes:

- Independent Lustre and IBM Spectrum Scale (GPFS) file systems for high-throughput, distributed storage.
- Dedicated Slingshot network fabrics for low-latency, high-bandwidth interconnectivity among nodes.
- Network Address Translation (NAT) gateways to manage outbound traffic flows and maintain security boundaries.
- Localized PBS Pro job scheduler instances to manage resource allocation and workload orchestration.
- Lightweight HPE Performance Cluster Management (HPCM) systems for baseline system orchestration and support.

This dual-zone configuration ensures operational resilience, fault tolerance, and the ability to support geographically distributed workloads if required.

Quad Structure and Node Management

Within each zone, compute node management is further subdivided across two instances of HPE Cray System Management (CSM), each responsible for managing approximately 2,700 nodes. Collectively, across the two zones, there are four CSM-managed node groups, referred to as "Quads."

The "Quad" architecture was intentionally designed to introduce additional security perimeterization and to create smaller, independent failover domains. In the event of a localized hardware, software, or network failure affecting one Quad, the remaining Quads continue to operate independently, minimizing overall system disruption. This subdivision also allows for targeted administrative actions, system updates, and troubleshooting without impacting the entire supercomputing environment.

Network Partitioning within Quads

Each Quad implements four network partitions to control and isolate network traffic among different types of nodes and services. The partitions are structured as follows:

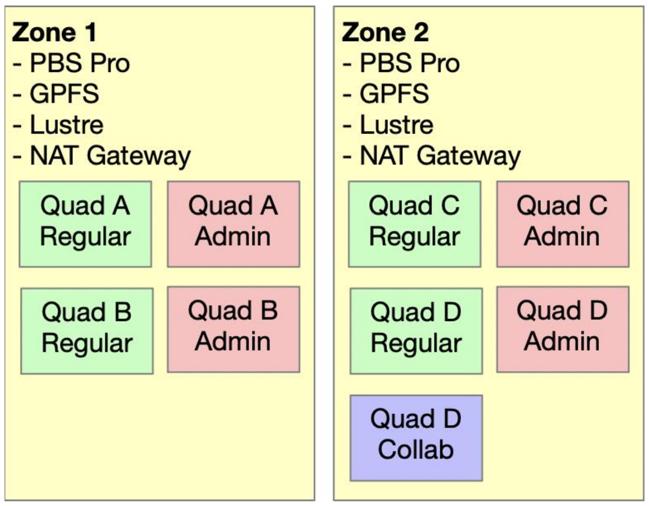
- **Admin Partition:** Dedicated to system administration traffic. This partition provides access to management services, monitoring systems, provisioning interfaces, and other privileged functions, isolated from general user traffic.
- **Regular Use Partition:** Supports all standard user activity, including job execution, file system access, and inter-node communication necessary for scientific and engineering workloads.

Additional segmentation exists within one of the Quads to accommodate users with lower security clearance levels:

- **Collaboration (Collab) Partition:** Created specifically for external collaborators or users whose workloads and data sensitivity do not meet the criteria for higher clearance environments. This partition provides controlled access to designated compute and storage resources, minimizing risk to sensitive operations elsewhere in the system.

Each network partition—across every Quad and every zone—has corresponding gateway routes explicitly defined within Microsoft Azure. These routing configurations ensure that each partition's traffic can be selectively routed and managed, maintaining strict separation and control across different network and security domains. The integration with Azure enables centralized oversight, scalable routing policy enforcement, and cloud-based security augmentation for external access management.

Image 3.1.1 - Diagram depicting topology and partitions



3.2 Configuration Management

Software Recipe Versions

The supercomputing environments deployed across the two zones were primarily based on the software recipes provided by HPE Cray System Management (CSM) version 1.4.1 and HPE Performance Cluster Management (HPCM) version 1.8. While each platform operates independently, it was essential to ensure that both maintained a consistent network topology and system behavior, resulting in a functional and cohesive gestalt across the entire distributed system.

Unified Configuration Management

To maintain alignment across all environments, an overarching configuration management schema was developed. This schema defined the authoritative source of truth for system configurations, spanning both CSM and HPCM-managed infrastructure.

Given CSM's native integration with Ansible within the Cray Fabric Services (CFS) framework, Ansible was selected as the standard tool for applying infrastructure configuration management across both CSM and HPCM environments. This decision allowed for consistency in management practices, tooling, and automation workflows across the heterogeneous platforms.

The schema, written in YAML, defined all critical system attributes, including but not limited to:

- Storage mount points (e.g., Lustre and GPFS mount paths)
- Node role assignments (e.g., compute, admin, storage)
- Static IP address assignments
- VLAN tagging and network segmentation
- Sudoers policy configuration

- LDAP integration parameters
- IPtables firewall rules for network isolation

This centralized approach enabled coordinated configuration management while still respecting the operational differences between CSM and HPCM.

HPCM Environments

In the HPCM-managed environments, a localized Git repository housed the Ansible playbooks and roles required for system updates. Feature branch commits containing configuration changes were pushed directly into the environment-specific Git repository. Upon validation, these playbooks were executed against targeted nodes to:

- Update system packages.
- Apply new network configurations (IP and VLAN assignments).
- Modify node role-specific settings (e.g., security rules for collaboration nodes).
- Rebuild system images as needed.
- Reboot nodes to apply new run-time configurations.

Azure DevOps pipelines were leveraged to control the injection of validated Ansible content into the environment, ensuring traceability and version control of all configuration changes.

CSM Environments

In CSM-managed environments, the deployment workflow followed the CFS model:

- Updated Ansible-based configurations were pushed into the Version Control System (VCS) backing CFS.
- CFS configurations were updated to reflect the latest schema and role assignments.
- New bootable system images were constructed incorporating the updated configurations.
- Nodes were then reimaged and rebooted with the latest images, ensuring that all security and network configurations, including IPtables firewall rules, were properly applied.
- Special attention was given to collab (collaboration) nodes, which required isolation from the regular compute environment. This was enforced through IPtables policies embedded during image build time.

The combination of CFS in CSM and Ansible in HPCM created a flexible but tightly controlled method to uniformly manage configuration drift across thousands of nodes.

3.3 Mechanisms of Isolation

VLAN-Based Traffic Segmentation on the High-Speed Network

In the supercomputing environment, Virtual Local Area Networks (VLANs) were utilized as the primary method for logically isolating traffic between different node groups operating on the high-speed Slingshot fabric. Each node group—whether general compute, storage, administrative, or collaboration—was assigned to a dedicated VLAN.

This segregation at the Layer 2 level ensured that:

- **Compute nodes** operated within isolated communication domains, preventing cross-talk with administrative or storage networks unless explicitly permitted.
- **Collaboration nodes** with lower security clearance levels were restricted to their own VLANs, isolating them from more sensitive workloads running elsewhere in the system.
- **Storage nodes and file systems** (e.g., Lustre and GPFS servers) operated within VLANs that only select compute nodes could access, based on job and data clearance requirements.

Each VLAN was explicitly mapped to the appropriate network interfaces on the nodes, enforced both in the configuration schema and at the switch fabric layer. This structure prevented unauthorized lateral movement between different functional domains on the high-speed interconnect and reduced the attack surface available within any individual VLAN.

VLAN Configuration at the Switch Port

One gap that still exists in this approach, given the limits of time, is that switchport VLANs are still managed by hand. If more time is allotted, the configuration schema could be extended to include this.

The HPE Aruba team has created a series of Ansible modules(3) for this purpose, including a module to create VLANs and assign them to switch ports(4).

Example 3.3.1 - Ansible inventory and playbook to create a VLAN 300 and assign it to a port j26.

```
Ansible Inventory

all:
  hosts:
    leafswitch_1:
      ansible_host: sw-leaf-bmc-001
      ansible_user: admin
      ansible_password: password
      ansible_connection: network_cli # SSH
connection
      method ansible_network_os:
      arubanetworks.aos_switch.arubaoss # Do not
change

--- Ansible Play

-  hosts: leafswitch_1
       collections:
         - arubanetworks.aos_switch
       tasks:
         - name: create vlan 300
             arubaoss_vlan:
               vlan_id: 300
               name: "vlan300"
               status: "VS_PORT_BASED"
               vlantype: "VT_STATIC"
               config: "create"
               command: config_vlan

        - name: assign vlan 300 to port 21
            arubaoss_vlan:
              vlan_id: 300
              port_id: 21
              command: config_vlan_port
```

CSM includes a script to automatically generate the Ansible inventory for switches(5).

IPtables-Based Access Control on the Management Network

While VLANs provided logical segmentation on the high-speed fabric, further protections were required on the lower-bandwidth management networks to secure administrative interfaces and configuration services.

To enforce security boundaries at the operating system level, custom IPtables firewall rules were configured and applied to all nodes as part of their runtime images. These rules were centrally generated based on node role, security clearance, and VLAN membership, and included policies such as:

- **Default-Deny Policy:** All inbound traffic to management interfaces was blocked by default, with explicit allow rules for trusted management sources (e.g., provisioning servers, LDAP servers, monitoring systems).

- **Role-Based Access:** Only nodes with administrative roles could initiate management protocol communications (e.g., SSH, PXE boot, image updates).

- **Clearance-Based Isolation:** Collaboration nodes had additional IPtables rules restricting their ability to communicate with nodes outside their designated partition, preventing unauthorized access to sensitive system components and high-security workloads.

- **Logging and Auditing:** IPtables was configured to log any denied traffic attempts, providing an audit trail for security monitoring and incident response.

The IPtables policies were embedded into the node images during the image build process (in CSM environments) or dynamically applied via Ansible playbooks (in HPCM environments), ensuring that enforcement was consistent and automatic across all nodes upon deployment or reboot.

Combined Effect

The combination of VLAN-based traffic isolation on the Slingshot high-speed network and IPtables-based access control on the management network provided a defense-in-depth architecture. VLANs contained and routed traffic appropriately at the network infrastructure level, while IPtables enforced additional node-level security policies, creating multiple independent layers of protection against unauthorized access, lateral movement, and misconfigured services.

Together, these strategies ensured that the multi-zone, multi-quad supercomputing system not only met but exceeded required security, availability, and compliance standards for both general-purpose and sensitive computing workloads.

3.4 Results

The distributed, multi-zone supercomputing system delivered several key results for the customer:

- **Operational Resilience:** Through architectural segmentation (zones, quads, VLANs, partitions), the system achieved strong fault isolation. A failure in one Quad or zone did not cascade into others, significantly reducing potential downtime and service disruption.

- **Security Compliance:** Network partitioning, VLAN isolation, IPtables enforcement, and role-based access controls ensured the system passed rigorous security audits. Sensitive data and jobs were isolated from lower-trust users and external collaborators, satisfying multilevel security (MLS) and internal data protection standards.

- **Performance Optimization:** Slingshot network segmentation enabled high-bandwidth, low-latency communication within trusted node groups while minimizing cross-domain traffic congestion. This helped maintain deterministic performance for critical workloads even under system-wide load.

- **Management Scalability and Efficiency:** Unified Ansible-driven configuration management and CFS workflows allowed rapid, consistent updates across thousands of nodes. Azure DevOps pipelines enabled version control, traceability, and reduced manual configuration errors.

- **Flexibility for Future Growth:** The YAML-based centralized schema simplified the onboarding of new nodes, partitions, and roles without re-architecting the system. Future expansion into additional zones or node groups could be absorbed without major design changes.

6 Conclusion

Dynamic network partitioning and flexible tenant management are now critical capabilities for secure, high-performance computing environments. As demonstrated across Slingshot, CSM, HPCM, and associated technologies, it is possible to achieve robust multi-tenancy, secure data segregation, and dynamic resource reallocation without compromising performance or operational efficiency.

Through techniques like VLAN-based traffic isolation, VNI enforcement, automated node reprovisioning, and defense-in-depth with IPtables, HPC systems can securely host mixed-sensitivity workloads, enable external collaboration, and remain adaptable to evolving organizational needs. The case study validated that with a careful combination of distributed architecture, automated configuration management, and tightly controlled network partitioning, even the most complex supercomputing environments can meet stringent security, scalability, and resilience requirements.

Looking forward, the integration of flexible, version-controlled network and system segmentation techniques will be foundational for future HPC deployments—enabling organizations to securely and efficiently serve diverse workloads, accelerate scientific discovery, and remain compliant with evolving security standards.

ACKNOWLEDGMENTS

ChatGPT was used to scaffold some of the basic content structure. Almost all of it was rewritten and restructured to provide the detail, but some phrasing may remain. Grammarly was used for grammar checking and simplification.

REFERENCES

[1] https://github.com/Cray-HPE/docs-csm/blob/release/1.7/operations/multi-tenancy/Tapms.md#webhook-payload

[2] Brewer, D.F.C.; Nash, M.J. (1989). "The Chinese Wall security policy" (PDF). Proceedings. 1989 IEEE Symposium on Security and Privacy. IEEE. pp. 206–214. doi:10.1109/SECPRI.1989.36295. ISBN 0-8186-1939-2. S2CID 7882054

[3] https://github.com/aruba/aos-switch-ansible-collection

[4] https://github.com/aruba/aos-switch-ansible-collection/blob/master/plugins/modules/arubaoss_vlan.py#L96

[5] https://github.com/Cray-HPE/docs-csm/blob/release/1.7/operations/network/management_network/canu_ansible_inventory.md?plain=1