Software-defined Multi-tenancy on HPE Cray EX Supercomputers

Jeremy Duckworth
Hewlett Packard Enterprise
jeremy.duckworth@hpe.com

Vinay Gavirangaswamy
Hewlett Packard Enterprise
vinay.gavirangaswamy@hpe.com

David Gloe
Hewlett Packard Enterprise
david.gloe@hpe.com

Brad Klein
Hewlett Packard Enterprise
bradley.klein@hpe.com

***Abstract*— Sandia National Laboratory's Red Storm System was designed to support "switching" hardware to isolate computation and data between data classification levels. This enabled Sandia and derivative system architectures to adapt investments in capability computing to evolving needs. Today, industry demand for multi-tenancy in modern converged HPC and AI platforms has not waned, but expectations around how the solution should be delivered have changed – as have the types of workloads being run. The industry is now strongly advocating for and investing in cloud-like platforms that treat multi-tenancy as a first-principles capability, align with modern DevOps management techniques, support resource elasticity, and enable customers to deliver their own IaaS, PaaS, and SaaS solutions. Enter HPE Cray Systems Management (CSM). CSM is a Kubernetes-based, turnkey, open source, API-driven HPC systems software solution. Using CSM as a foundation, we have developed a software-defined, multi-tenancy architecture, anchored by a tenancy "controller hub," called the Tenant and Partition Management System (TAPMS). TAPMS, through extant features in CSM, inherits availability, scale, resiliency, disaster recovery, and security properties of the platform. This paper presents TAPMS, the supporting architecture, and the resulting composable, declarative tenant configuration interfaces that TAPMS and the underlying Kubernetes Operator Pattern enable.**

Keywords—Multi-tenancy, multitenancy, high performance computing, HPC

I. Introduction

Supercomputers are increasingly employed to host unconventional workloads alongside HPC simulations, incorporating computational methods like AI and ML into more sophisticated workflows, some spanning multiple distributed but discrete systems. Success with these workflows ultimately necessitates accommodation of their runtime requirements and configuration management practices, catalyzing some in the industry to push the boundary of HPC system solutions to adopt modern platform engineering techniques [3, 4]. By integrating *key* cloud native technologies and management techniques with those of traditional supercomputing, purposefully reconfigurable HPC system solutions are beginning to emerge [5, 6, 20] outside of public cloud. These systems have an opportunity to exploit the data performance and data locality efficiencies inherent in supercomputing with configuration flexibility only found in the cloud, today [3]. While the origins of multi-tenancy in HPC date at least back to Sandia National Laboratory's Red Storm System [1, 2], and hardware "switching", cloud technologies have revolutionized the space. Cloud native multi-tenancy architectures rely heavily on rapid, elastic, logical reconfiguration of resources through well-defined software interfaces – representing a significant advancement in programmable infrastructure.

In this paper, we start by defining multi-tenancy, its differentiating characteristics, and related terms (section II). We then explore and summarize strengths and weaknesses associated with multi-tenancy (section II). Next, we briefly discuss the "as a service" (aaS) business model, practically inseparable from a discourse on multi-tenancy (section II). After providing background, we describe our work (section III) on development of a software-defined multi-tenancy feature set in Cray Systems Management (CSM) using Cray EX Hardware. In closing, we provide a conclusion (section IV) and cover future work (section V).

II. Background

A. *Multi-tenancy Definition* and *Related Terms*

We adopt a definition of multi-tenancy from [15]:

"Multi-tenancy is a property of a system where multiple customers, so-called tenants, transparently share the system's resources, such as services, applications, databases, or hardware, with the aim of lowering costs, while still being able to exclusively configure the system to the needs of the tenant."

While multi-tenancy is intuitively a sharing of resources, the first distinguishing characteristic of note here is that resources are shared in a transparent manner – meaning tenants are unaware of other tenants and view their own resources as a dedicated environment [16]. The second distinguishing characteristic is that the system can be exclusively configured to meet the needs of the tenant. In contrast, while multi-user systems can be configured to accommodate several classes of users, they are often unable to cater exclusively to each class individually. Much like a multi-user environment, configurability also includes reserving and then guaranteeing performance and other quality of service measures [17].

Multi-instance and elasticity are terms that are often concomitant with multi-tenancy. Multi-instance is a term used to describe the scenario where there is no sharing of resources between tenants. For example, rather than multiple tenants using a single application instance in a SaaS environment, they each have access to independent copies or instances [16]. Elasticity is a system property whereby resources can be rapidly [23] added or removed from a tenant's resource pool to address changes in workload demand and manage cost [17, 27].

B. Multi-tenancy: Strengths and Weaknessnes

Some strengths of multi-tenancy are in lowering costs for either the consumer or provider [15] through leveraging economies of scale [15] or gains in energy efficiency (power savings) [17]; and accelerating availability of the latest technology [18]. Elasticity is a related but distinct strength often associated with multi-tenancy, whereby a tenant can scale resources based on workload demand [17, 27]. While isolation of data and processing between tenants is innate in multi-tenancy, some multi-tenant systems are purposefully designed to support hard multi-tenancy. Hard multi-tenancy requires strict isolation between tenants, such as might be required in a multi-level security (MLS) environment [1, 19]. Conversely, soft multi-tenancy is optimized for accident prevention (e.g., avoiding outages) and agility versus strict isolation [24].

Some strengths may skew towards the HPC industry. The first is using multi-tenancy to isolate heterogenous workloads while still benefiting from the data locality and accelerated performance of the underlying platform [3]. With elasticity and rapid reconfiguration, the system could be decomposed into smaller tenants to manage heterogenous software and hardware workflows, and then quickly reassembled into a capability supercomputing asset to run jobs at scale, and vice-versa, making supercomputing systems more adaptable. When combined with DevOps practices, multi-tenancy in HPC could also be deployed to serve select features of a Test and Development System (TDS), including support of a "blue green" deployment model [25], where configuration changes could be easier to orchestrate across tenants versus across independent systems.

Potential weaknesses associated with multi-tenancy are increased software complexity [16], difficulty in managing change without impacting multiple tenants [16] and heightened cybersecurity concerns (e.g., a heightened sense of urgency to quickly patch security vulnerabilities) [16]. As cybersecurity in HPC is often optimized towards maximum performance versus robust security [26], it could be considered a weakness if not treated properly in a multi-tenant design. **Table I** summarizes strengths and weaknesses.

C. As a Service (aaS) Business Model

Multi-tenancy is often associated with the as a service (aaS) business model as a key enabler [16, 17]. Services are typically delivered across stratified layers (**Table II**), and service providers often leverage capabilities in lower layers (e.g., IaaS) to easily offer high order services (e.g., WaaS) [17].

Operational risks are amortized across all parties involved in delivery or consumption of the service. Parties typically require contractual agreements to protect shared interests and limit liability should events like a security breach or data loss occur, often canonized into a shared responsibility model [28].

Strengths		***Weaknesses***	
General	***HPC***	***General***	***HPC***
Cost Savings (provider, consumer) via shared hardware, economies of scale, and energy efficiency Elasticity Accelerated access to the latest technology Support for multiple security trust domains per system to meet MLS requirements	Rapid, software-based reconfigurability, at very large scale Logical Test and Development Systems, support for blue/green deployments	System Complexity Difficulty in managing change without impacting multiple tenants	Multi-tenancy requires "advanced" security

TABLE I. MULTI-TENANCY: STRENTGHS AND WEAKNESSES

Layer	***Description***
Workflow as a Service (WaaS)	Hosted workflows, like those that can be modeled as Directed Acyclic Graphs (DAGs) [17]
Function as a Service (FaaS)	Hosted functions [22]
Database as a Service (DBaaS)	Hosted databases [17]
Software as a Service (SaaS)	Hosted applications [17]
Platform as a Service (PaaS)	Hosted operating systems [17]
Infrastructure as a Service (IaaS)	Hosted virtual machines or bare metal services [17]

TABLE II. AS A SERVICE LAYERS

III. MULTI-TENANCY: CRAY SYSTEMS MANAGEMENT

HPE Cray CSM (Cray Systems Management) is a cloud native HPC platform solution [6], based on the Shasta architecture [5]. CSM is open-source software (MIT), first released as closed source approximately two years ago, and is soon to celebrate its fifth major release (1.4). CSM currently supports HPE Cray EX hardware and provides a base platform for several hardware and software product streams comprising a complete HPC solution, such as the Cray Operating System (COS), the System Monitoring Application (SMA), the Cray Programming Environment (CPE), and the Slingshot High-speed Network (HSN) Fabric. CSM represents a notable paradigm shift in systems architecture and is being successfully hardened by the first cohort of supercomputers it has been deployed to manage.

CSM heavily leverages Kubernetes (K8s) [10], along with a larger portfolio of Cloud Native Compute Foundation Landscape [33] and other open technologies. Management services in CSM run as micro-services inside K8s, using Istio [29] for secure service-to-service communication, and build

upon zero downtime features rooted in K8s. CSM further leverages Keycloak (for federated person identities) and Spire/SPIFFE [31] (for non-person identities) to provide horizontally scaled, Zero Trust Architecture [30] focused API and UI ingress services. CSM uses Ceph [32] for remote block, clustered file system, and S3-compatible utility storage. Presently, all deployed services in the management and compute zones [26] execute on bare metal[1], except for an emerging feature that uses virtualization techniques to emulate ARM OS image builds on x86_64 K8s worker nodes. An investigation is also currently under way to virtualize K8s master and worker nodes towards online migration between management clusters, and support for "blue green" promotion techniques [25].

A. Community Engagement

The Swiss National Supercomputing Centre (CSCS) and HPE Cray are collaborating on the CSM multi-tenancy feature set. CSCS has a history of innovation in the space [3], and eventually plans to host several, production grade tenants on their Alps system [34] – through a vision CSCS calls "software-defined infrastructure." In 2023, CSCS and HPE Cray started an open, co-chaired multi-tenancy special interest group (SIG). Over the course of approximately the last year and a half, CSCS and HPE Cray have worked to accelerate the multi-tenancy feature set in CSM.

B. Design Philosophy and Approach

Multi-tenancy is currently being designed as a feature toggle that must peacefully coexist with and ideally improve the incumbent, single-tenancy user experience. Consequently, all assignable resources that are not explicitly associated with a named tenant belong to an implicit infrastructure tenant.

Multi-tenancy in CSM is focused on PaaS-aligned, soft multi-tenancy. We plan to iterate towards "harder" multi-tenancy through key isolation capabilities. In an attempt to balance implementation complexity, amortize risk to the overall product, and make efficient use of existing management system hardware, tenant awareness is being phased into CSM management API micro-services. For tenant owned or configurable resources, we are focused on bare metal application and compute nodes, and the software resources associated with their configuration and operation.

As a configuration approach, we are aligning with DevOps and GitOps patterns [35], in some cases using these patterns as a transition path towards a completely native, multi-tenancy offering in CSM. To support tenant-driven configurability, we have introduced a new CSM IAM (Identity and Access Management) persona, that of tenant administrator, and re-positioned the existing global administrative persona as infrastructure administrator. While we are approaching the tenant administrator model from a least-privilege perspective, we are also accumulating architectural guidelines for development of the access control model, such as:

- Tenant administrator access to hardware control or configuration **must** be constrainted through API middleware (e.g., no direct access to RedFish [40] endpoints)
- For power control, tenant administrators **must** be limited to nodes in the power hierarchy (not slots, chassis, etc)
- Tenant administrators **must not** have the ability to configure power capping
- Tenant administrators **must not** have the ability to update firmware.

Development has progressed across phases involving multiple HPE Cray product streams. The first phase was started and completed in 2022. Feature content for phase two was defined earlier this year, with implementation starting in April of 2023. Planning for our phases is expected to start later this year. We detail the user stories, objectives, and design for phases one and two in the next section.

C. Soft PaaS Multi-tenancy: Phase One

The theme of phase one was the incorporation of CSCS's "virtual cluster" [3] design into the CSM product, with a desired outcome of yielding tighter integration, better automation, and a progressive shift towards a GitOps development experience. We also established initial IAM extensions for administrative tenant use cases. The resultant capabilities were delivered as a technical preview in the 22.11 Software Release, which included CSM 1.3. Each user story (US) included in phase one (verbatim) is listed below, accompanied by a feature response.

US1: As an infrastructure administrator, I would like to create, update, and delete tenants.

Infrastructure administrators can manage tenants via the Tenant and Partition Management System (TAPMS) [7]. TAPMS is the "controller hub" of CSM's composable, declarative, multi-tenancy architecture. TAPMS is implemented as a K8s Operator [8]. Thus, the principal API (Application Programming Interfaces) for TAPMS is the K8s API, and tenant provisioning occurs based on the K8s operator reconciliation process. The Kubernetes Custom Resource Definition (CRD) data model provides tenant schema validation, enables support of multiple schema versions, and enables support of schema version migration. Custom resources, much like native K8s resources can also be stored in git as a YAML or JSON file. This also makes tenant definitions suitable for management by GitOps focused continuous deployment (CD) tools like Argo CD [12] or Flux CD [13].

TAPMS is integrated with the Hierarchical Namespace Controller (HNC) [9] to provide namespace-based multi-tenancy in K8s. As tenants are created, TAPMS, through HNC, creates a hierarchical tree of K8s Namespaces[2] based on the declared configuration. These namespaces could be used to host

[1] Management services are heavily containerized.

[2] K8s does not natively support namespace hierarchies.

siloed "management" software instances, such as Slurm. Creation of child namespaces in this way is optional.

Through IAM (Identity and Access Management) extensions, TAPMS is integrated with the Keycloak [11] OpenID Connect (OIDC) identity provider in CSM, where it provides one group per tenant, and automates the group mapping to select OIDC clients in the "shasta" realm. Infrastructure administrators can then assign these groups to a tenant administrator in Keycloak, and the resulting credentials can be used for federated K8s administration (e.g., via kubectl). By default, no users are assigned to tenant groups, and no access exists for federated users. The capability was initially added to support the use case whereby infrastructure administrators opt to allow tenant administrators patently restricted access to a workload running in a tenant namespace, like viewing logs in Slurm [37] K8s pods. It will be extended to support API gateway access control as part of phase two.

TAPMS schedules compute node resources into a tenant through integration with Hardware State Manager (HSM) in CSM, using HSM exclusive groups. Exclusive groups enforce partitioning behavior at the node set level. Nodes can be added or removed from the TAPMS tenant configuration providing rudimentary elasticity support. Internally, TAPMS tracks a unique UUID for each tenant instantiation (separate from K8s UUID on the resource).

Figure 1 illustrates the TAPMS tenant provisioning process. As TAPMS is reconciling the state of a tenant, it updates the tenant custom resource with a status of *New, Deploying, Deployed, or Deleting.* **Figure 2** illustrates an example TAPMS tenant configuration file. See [14] for CSM's multi-tenancy documentation, in CSM 1.3.

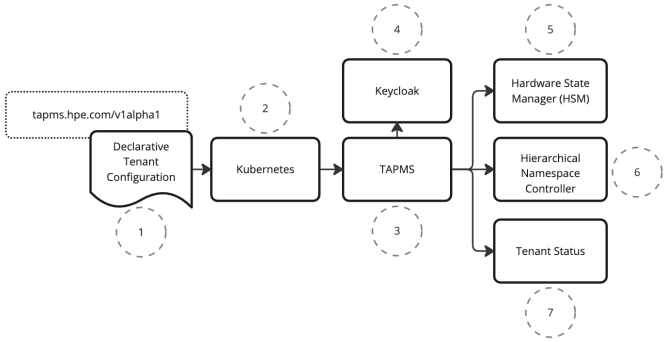

Fig. 1. TAPMS Tenant Provisioning

US2: As an infrastructure administrator, I would like a clear procedure for configuring UANs and CNs with their respective Slurm credentials and LDAP configuration.

After a tenant specific Slurm cluster is deployed as subsequently described in **US3**, Ansible (CFS) Group Variables, matching the HSM exclusive group, can be used to deploy the configuration to application or compute nodes as described in the documentation tree at [14].

```yaml
apiVersion: tapms.hpe.com/v1alpha1
kind: Tenant
metadata:
  name: vcluster-blue
spec:
  childnamespaces:
    - slurm
  tenantname: vcluster-blue
  tenantresources:
    - type: compute
      hsmgrouplabel: blue
      enforceexclusivehsmgroups: true
      xnames:
        - x0c3s5b0n0
        - x0c3s6b0n0
```

Fig. 2. TAPMS Tenant Declaration

US3: As an infrastructure administrator, I would like to not have jobs or nodes exposed across tenants. Likewise, I would like the availability of tenant resources to be protected.

Infrastructure administrators can automate the deployment of one Slurm cluster per tenant using the HPE Slurm K8s Operator. The Slurm operator is configured in a similar manner as TAPMS also references the TAPMS tenant resource (the operators are composable). The Slurm operator sources node membership from TAPMS and automatically reacts to changes in TAPMS node membership (as nodes are added or removed). As only nodes that belong to a tenant are accessible in the multi-instance model of Slurm, jobs and nodes are not exposed by Slurm outside the tenant. Usage here is not related to Slurm native multi-tenancy features.

In the case of the example Slurm Operator configuration shown in **Figure 4**, the Slurm server-side components (slurmctl, slurmdb, pxc, etc.) are deployed into an HNC child namespace previously provisioned by TAPMS. **Figure 3** illustrates the Slurm Operator tenant provisioning process.

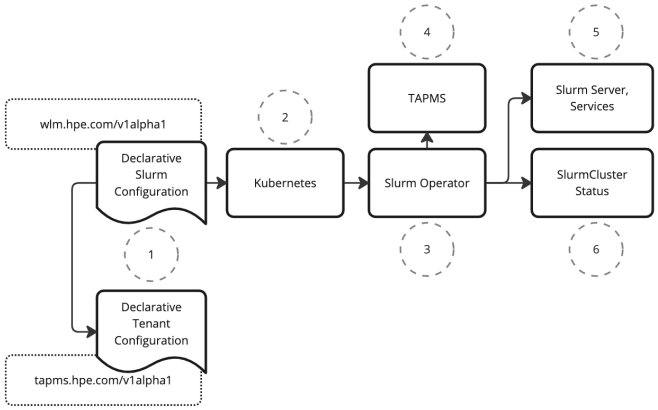

Fig. 3. Slurm Cluster Tenant Provisioning Process

```yaml
apiVersion: "wlm.hpe.com/v1alpha1"
kind: SlurmCluster
metadata:
  name: mycluster
  namespace: vcluster-blue-slurm
spec:
  tapmsTenantName: vcluster-blue
  tapmsTenantVersion: v1alpha1
  slurmctld:
    ...
```

Fig. 4. Slurm Cluster Tenant Declaration

US4: As a tenant user, I would like to log into my user environment using my tenant-specific credentials and orchestrate Slurm jobs against the nodes in my tenant compute partition.

After (UANs) are provisioned using the identity provider of choice, the Slurm configuration can be deployed as noted in response to **US2**.

D. Soft PaaS Multi-tenancy: Phase Two

The themes of phase two are enablement of the tenant administrator persona for basic image build, node configuration, and node boot use cases, and tenant network isolation. Each user story currently included in phase two is listed below, accompanied by a working design response.

US5: As a tenant administrator, I would like to build and customize compute, user access, and application node images, limited to the tenants that I manage.

The set of CSM micro-services and APIs that would need to be refactored to accomplish this using native CSM multi-tenancy awareness is large, would impact several critical path services, and would likely have considerable downstream effects that could slow overall product velocity. As an alternative to refactoring a large dependency tree in parallel and the associated risk, we are currently planning to enable tenant administrator functionality here via a prevailing DevOps pattern. Specifically, we plan to support mirroring of the CSM Gitea VCS for configuration state into an enterprise Git implementation, external to CSM. There, access controls could be put into place that appropriately reflect the providers shared responsibility practices. In the canonical tenant administrator case, a change would first have to be approved by an infrastructure administrator, be further subjected to other conditions and checks before merge, and then be provisioned indirectly through a pipeline build agent that might also execute system tests for further validation. **Figure 5** illustrates the DevOps pattern for tenant administration. The red lines represent privilege control boundaries. As allowing tenants to manipulate operating system image configurations begins to blur the line between PaaS and IaaS, and care must be taken not to overexpose IaaS layers to tenants, due acutely to cybersecurity related risks that may arise. The provisioning tools in scope, notably Ansible and CFS, have broad authority at the infrastructure layer in CSM. Note that the default state of the VCS repository structure and provisioning logic does not innately support multi-tenancy currently – requiring site customizations and perhaps custom Git project structures. This is an area of active discussion and collaboration but will likely remain an integration exercise for the foreseeable future.

We also plan to add two automation focused features into TAPMS. The first is basic power fencing when a node is added to or removed from a tenant, which will guard against volatile state leakage. The second is a configurable webhook capability, where webhooks can be configured as blocking or non-blocking and will exist for common types of configuration events (e.g., additions or removals of nodes). Webhooks will be configured as part of the TAPMS tenant declaration. We plan to incorporate their use into CSM services to act as barriers for gating tenant transitions where needed (e.g., don't allow a node to transition in the middle of a firmware upgrade operation), and as a light-weight messaging model. We anticipate their use outside of CSM as well for similar use cases, and this is one reason we are pursuing this functionality in a webhook form rather than a message bus. See **Figure 6** for an illustration of the webhook, application node tracking, and power fencing concept.

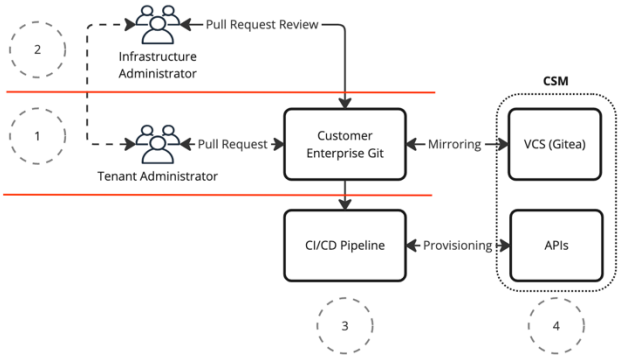

Fig. 5. DevOps Pattern for Tenant Administration

US6: As a tenant administrator, I would like to automate post-boot configuration of compute, user access, and application nodes, limited to the tenants that I manage.

We are planning to use a similar approach to the one noted in **US5**, with one exception – secrets management. For a tenant-aware secret management solution, we are currently researching the use of encrypted secrets in Git, protected by per-tenant asymmetric cryptographic keys in Hashicorp Vault, using Mozilla's SOPS (Secrets as Operations) [39] as both an Ansible (CFS) plugin and interactive CLI tool. In this model, TAPMS would provision a tenant specific Hashicorp Vault Transit engine and key(s) during tenant provisioning and expose public key material that could be shared with tenant administrators. Then, during CFS execution, Ansible (AEE) would make API calls into the Vault Transit Engine [38] requesting decryption, but not exposing private key material. Provisioning per-tenant key management service (KMS) enclaves could also be used for future use cases, as well.

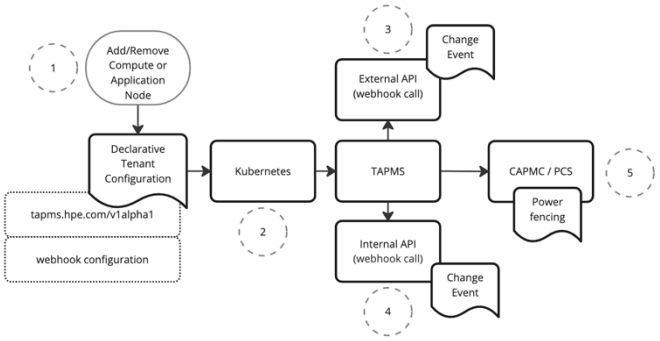

Fig. 6. TAPMS webhook, application node tracking, and power fencing concept

For rudimentary node targeting, HSM exclusive groups can be used via Ansible dynamic inventory, where these groups are managed by TAPMS. Application nodes (e.g., UANs) are also being added to TAPMS as resources that can be assigned to a tenant.

US7*:* As a tenant administrator, I would like to assign bootable images, kernels, kernel parameters, initrd, and CFS plays to compute, user access, and application nodes, limited to the tenants that I manage.

We are planning to use a similar approach to the one noted in **US5**. In this case, a configuration set (in git) that expresses the desired configuration state and CI/CD pipelines to reconcile state is additionally required.

US8: As a tenant administrator, I would like to observe the status of image build and node boot operations, limited to the tenants that I manage.

We are planning to use a similar approach to the one noted in **US5**. Observability for at least image build operations may initially be limited to CI/CD pipeline-based operations and agent job output.

US9: As a tenant administrator, I would like to power nodes on and off and perform OS-level shutdown and reboot operations on compute, user access, and application nodes, limited to the tenants that I manage.

For this user story, we plan to focus on implementing the noted functionality using native tenant awareness in the Boot Orchestration Service (v2), as an exemplar service. We plan to leverage lessons learned from this experience to establish a blueprint for transitioning other services and as a baseline for future system architectures. Tenant administrator access will be limited to the discrete operations listed, and configuration of the BOS boot "session templates" and related resources will be addressed as previously discussed.

To implement tenant awareness in BOSv2, we plan to add tenant awareness to the CLI and API gateway implementations in CSM. A configuration context setting will be added to the Cray CLI to support configuration of a tenant name. Then, when the configuration context is active, the Cray CLI will add an associated header to all API requests. After the API gateway (Istio) receives the request, it will pass the request through Open Policy Agent (OPA), where OPA will apply the following rules (in addition to fundamental processing like cryptographic validation of the bearer token [40]):

1. If the request contains a tenant name header, and the role is infrastructure administrator, match the request attributes against the API access control list (e.g., allowed BOS HTTPS endpoints and HTTP verbs) that applies to the infrastructure administrator role.
2. If the request contains a tenant name header, and the role is tenant administrator, first validate that the bearer token contains a group claim that grants the tenant administrator access to the tenant. Then, match the request attributes against the API access control list that applies to the tenant administrator role.

If the request is authenticated and authorized, then it is allowed to pass to BOS. Once BOS receives the request, to validate that the scope of the request is within the bounds of the tenant, it will call out to a read-only, API endpoint in TAPMS for tenant status and composition. Once it contextually validates the request via interactions with TAPMS, it will attempt to service the request. In addition to this behavior, the BOS team is currently investigating application name spacing techniques to partition BOS managed application objects for tenant ownership. In this case, if an infrastructure administrator wanted to interact with tenant owned resources, the request would need to be scoped to the tenant (e.g., via the Cray CLI setting). **Figure 7** illustrates the request flow described above. Cray CLI, Istio, OPA, and KeyCloak (OIDC as bearer token implementation) are pre-existing components in CSM that we are building upon.

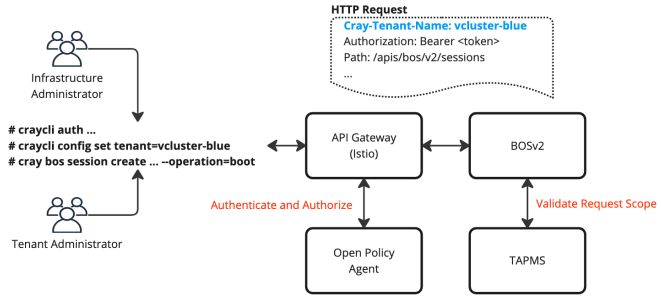

Fig. 7. Tenant aware BOSv2 API Request

US10: As an infrastructure administrator, I would like my node management network and HSN networks to be logically isolated, so that there is no cross-tenant network communication.

For the NMN (node management network), we are planning to support a static configuration that prevents managed node to managed node communication, while also restricting managed node to management node communication (isolated to only required ports and protocols). For Cray Ex Mountain hardware,

isolation will be limited to units larger than a chassis, and the underlying control is accomplished via Layer 3 and 4[3] NACLs (network access control lists) implemented in the CDU switch pair. For Cray Ex River hardware, isolation would be limited to application nodes, and the current plan is to isolate each application node into its own Layer 3 network, using a point-to-point configuration that masquerades as the existing, canonical NMN network address (to provide backward compatibility). Support for this configuration may be limited to Aruba switch hardware, and we may decide to feature toggle this functionality. The HMN (hardware management network) is not being partitioned as it is an OOB (out of band) network, and there are no plans to support tenant presence in this network enclave.

The Slingshot team is planning to deliver an operator, similar in form to the TAPMS and Slurm operators previously covered, that integrates with at least TAPMS for node topology and tenant state, and then uses fabric-managed, switch-enforced policies to logically isolate traffic within a shared Slingshot fabric at tenant boundaries. For VNIs, the operator will allocate a block for the tenant, that can further be sub-allocated as needed (e.g., for Slurm). Plans for the operator also include the ability to provision shared VNIs and accommodate VLAN-based isolation policies. See **Figure 8** for an illustration of Slingshot tenant provisioning.

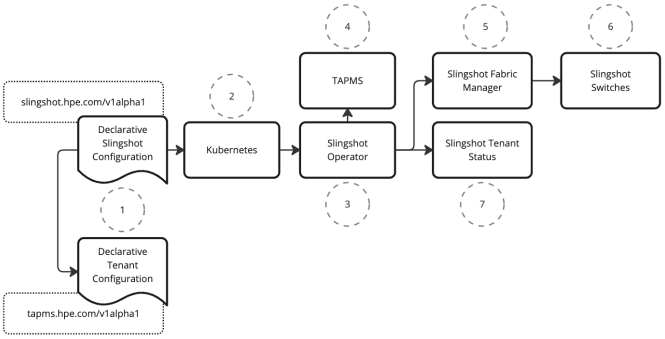

Fig. 8. Slingshot Operator Tenant Provisioning Model

IV. Conclusion

Multi-tenancy, the aaS business model, and heterogenous HPC workflows are catalyzing the state of the art for programmable infrastructure and cybersecurity in HPC. HPE Cray, in partnership with CSCS and the HPC community, is engaged and actively influencing these trends towards improved outcomes in scientific computing. We are excited about the future of the technology and applications, and opportunities for collaborative development with the HPC community.

V. Future Work

Our immediate focus is on helping CSM users, like CSCS, to operationalize the phase one multi-tenancy capabilities, and likewise for phase two to meet production goals. Next, as the demarcation point between PaaS and IaaS may benefit from added clarity, we are exploring use case alignment, alongside technologies, designs, and trade-offs that could bring true IaaS multi-tenancy to large scale HPC. Finally, while multi-tenancy represents a very broad and diverse set of architectural concerns, we would like to explore predicate-based scheduling in TAPMS (e.g., implicit tenant resource selection by hardware properties versus explicit geolocation identifiers), and the state of HPC storage multi-tenancy from a systems architecture perspective.

Acknowledgment

We would like to acknowledge CSCS for their collaboration in this space, and specifically the following current and former colleagues (in no order): Mark Klein, Felipe Cruz, Miguel Gila, Cerlane Leong, Maxime Martinasso, Narendra Challa, and Sadaf Alam. We would also like to acknowledge others on the HPE Cray Team that have and continue to contribute towards forward progress and critical review and discussion (again in no order): Larry Kaplan, Jesse Treger, Ed Benson, Marten Turpstra, Jason Sollom, Andrew Nieuwsma, and Dennis Walker. Finally, we highly value the interactions we have had through the CSM Multi-tenancy SIG with technical leadership in the HPC community and desire to make this an increasingly more vibrant and active forum for future collaboration.

References

[1] Tomkins, James L. Red Storm: The Birth of a New Supercomputer. United States: N. p., 2008. https://www.osti.gov/biblio/1142434.

[2] Sandia Press Release. Sandia Red Storm supercomputer exists world stage. 2012. https://newsreleases.sandia.gov/red-storm-exits/.

[3] S. R. Alam, M. Gila, M. Klein and M. Martinasso, "Multi-tenancy Management and Zero Downtime Upgrades using Cray-HPE Shasta Supercomputers," 2021 SC Workshops Supplementary Proceedings (SCWS), St. Louis, MO, USA, 2021, pp. 87-94, doi: 10.1109/SCWS55283.2021.00021.

[4] B. S. Allen, M. Ezell, P. Peltz, D. Jacobsen, E. Roman, C. Lueninghoener, et al., "Modernizing the HPC System Software Stack", arXiv, 2020, [online] Available: https://arxiv.org/pdf/2007.10290.pdf.

[5] Cray HPE Shasta Software Stack. 2019. https://cug.org/proceedings/cug2019_proceedings/includes/files/inv113s1-file1.pdf.

[6] HPE Cray Systems Management (CSM) Open-Source Software. https://github.com/Cray-HPE/. Last accessed May 4, 2023.

[7] HPE Cray Tenant and Partition Management System (TAPMS) Kubernetes Operator. https://github.com/Cray-HPE/cray-tapms-operator. Last accessed May 4, 2023.

[8] Kubernetes Operator Pattern. https://kubernetes.io/docs/concepts/extend-kubernetes/operator/. Last accessed May 4, 2023.

[9] Hierarchical Namespace Controller. https://github.com/kubernetes-sigs/hierarchical-namespaces. Last accessed May 4, 2023.

[10] Kubernetes Overview. https://kubernetes.io/docs/concepts/overview/. Last accessed May 4, 2023.

[11] Keycloak: Identity and Access Management (IAM) Solution. https://www.keycloak.org/. Last accessed May 4, 2023.

[12] Argo CD: Declarative GitOps CD for Kubernetes. https://argo-cd.readthedocs.io/en/stable/. Last accessed May 4, 2023.

[3] Open Systems Interconnect (OSI) network and transport layers

[13] FLUX CD. HTTPS://FLUXCD.IO/. LAST ACCESSED MAY 4, 2023.

[14] MULTI-TENANCY SUPPORT, CSM. HTTPS://GITHUB.COM/CRAY-HPE/DOCS-CSM/BLOB/RELEASE/1.3/OPERATIONS/MULTI_TENANCY/OVERVIEW.MD. LAST ACCESSED MAY 4, 2023.

[15] KABBEDIJK, JAAP, ET AL. "DEFINING MULTI-TENANCY: A SYSTEMATIC MAPPING STUDY ON THE ACADEMIC AND THE INDUSTRIAL PERSPECTIVE." JOURNAL OF SYSTEMS AND SOFTWARE 100 (2015): 139-148.

[16] C. P. BEZEMER, A. ZAIDMAN, B. PLATZBEECKER, T. HURKMANS AND A. HART, "ENABLING MULTI-TENANCY: AN INDUSTRIAL EXPERIENCE REPORT". 2010 IEEE INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE, TIMISOARA, ROMANIA, 2010, PP. 1-8, DOI: 10.1109/ICSM.2010.5609735.

[17] JIA, RU, ET AL. "A SYSTEMATIC REVIEW OF SCHEDULING APPROACHES ON MULTI-TENANCY CLOUD PLATFORMS." *INFORMATION AND SOFTWARE TECHNOLOGY* 132 (2021): 106478.

[18] S. JAMALIAN, H. RAJAEI, ASETS: A SDN EMPOWERED TASK SCHEDULING SYSTEM FOR HPCAAS ON THE CLOUD, IN: PROCEEDINGS OF IEEE INTERNATIONAL CONFERENCE ON CLOUD ENGINEERING (IC2E), IEEE, 2015, PP. 329-334.

[19] NIST SP 800-53R5. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf.

[20] STACKHPC, LTD. HTTPS://WWW.STACKHPC.COM/PAGES/ABOUT.HTML. LAST ACCESSED MAY 4, 2023.

[21] D. SATO. CANARY RELEASE, MARTIN FOWLER BLOG. HTTPS://MARTINFOWLER.COM/BLIKI/CANARYRELEASE.HTML. LAST ACCESSED MAY 4, 2023.

[22] M. ROBERTS. SERVERLESS ARCHITECTURES (FAAS). MARTIN FOWLER BLOG. https://martinfowler.com/articles/serverless.html#unpacking-faas. LAST ACCESSED MAY 4, 2023.

[23] NIST SP 800-145. HTTPS://NVLPUBS.NIST.GOV/NISTPUBS/LEGACY/SP/NISTSPECIALPUBLICATION800-145.PDF.

[24] AWS RE:INVENT 2019: ARCHITECTING MULTI-TENANT PAAS OFFERINGS WITH AMAZON EKS (GPSTEC337). https://www.youtube.com/watch?v=P29eL_51iYU. LAST ACCESSED MAY 4, 2023.

[25] M. FOWLER. BLUE GREEN DEPLOYMENT. HTTPS://MARTINFOWLER.COM/BLIKI/BLUEGREENDEPLOYMENT.HTML. LAST ACCESSED MAY 4, 2023.

[26] NIST 800-223 INITIAL PUBLIC DRAFT. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-223.ipd.pdf.

[27] BUILDING THE #7 GLOBAL SUPERCOMPUTER WITH GOOGLE KUBERNETES, PETROLEUM GEO-SERVICES (PGS). https://www.pgs.com/company/newsroom/news/industry-insights--hpc-in-the-cloud/. LAST ACCESSED MAY 4, 2023.

[28] DIGITALOCEAN, SHARED RESPONSIBILITY MODEL. https://www.digitalocean.com/trust/faq. LAST ACCESSED MAY 4, 2023.

[29] ISTIO SERVICE MESH. https://istio.io/. LAST ACCESSED MAY 4, 2023.

[30] NIST ZERO TRUST ARCHITECTURE. https://csrc.nist.gov/publications/detail/sp/800-207/final.

[31] SPIRE/SPIFFE. https://github.com/spiffe/spire. LAST ACCESSED MAY 4, 2023.

[32] CEPH. https://docs.ceph.com. LAST ACCESSED MAY 4, 2023.

[33] CLOUD NATIVE COMPUTE FOUNDATION LANDSCAPE TECHNOLOGIES. https://landscape.cncf.io/. LAST ACCESSED MAY 4, 2023.

[34] INTRO TO ALPS: WHAT EXACTLY IS THE NEW SWISS SUPERCOMPUTER INFRASTRUCTURE? HTTPS://WWW.HPCWIRE.COM/2023/04/05/INTO-THE-ALPS-WHAT-EXACTLY-IS-THE-NEW-SWISS-SUPERCOMPUTER-INFRASTRUCTURE/. LAST ACCESSED MAY 4, 2023.

[35] HISTORY OF GITOPS. https://www.weave.works/blog/the-history-of-gitops. LAST ACCESSED MAY 4, 2023.

[36] GITLAB. https://about.gitlab.com/. LAST ACCESSED MAY 4, 2023.

[37] SLURM. https://www.schedmd.com/. LAST ACCESSED MAY 4, 2023.

[38] HASHICORP VAULT TRANSIT ENGINE. https://developer.hashicorp.com/vault/docs/secrets/transit. LAST ACCESSED MAY 4, 2023.

[39] MOZILLA SOPS. https://github.com/mozilla/sops. LAST ACCESSED MAY 4, 2023.

[40] REDFISH STANDARD, DISTRIBUTED MANAGEMENT TASK FORCE (DMTF). HTTPS://WWW.DMTF.ORG/STANDARDS/REDFISH. LAST ACCESSED MAY 4, 2023.