International Journal of
HIGH PERFORMANCE
COMPUTING APPLICATIONS

Special Issue Paper

The International Journal of High Performance Computing Applications
2023, Vol. 37(3-4) 288–305
© The Author(s) 2023
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/10943420231167811
journals.sagepub.com/home/hpc
Sage

Versatile software-defined HPC and cloud clusters on Alps supercomputer for diverse workflows

Sadaf R Alam[1], Miguel Gila[2], Mark Klein[2], Maxime Martinasso[2] and Thomas C Schulthess[2,3]

Abstract
Supercomputers have been driving innovations for performance and scaling benefiting several scientific applications for the past few decades. Yet their ecosystems remain virtually unchanged when it comes to integrating distributed data-driven workflows, primarily due to rather rigid access methods and restricted configuration management options. X-as-a-Service model of cloud has introduced, among other features, a developer-centric DevOps approach empowering developers of infrastructure, platform to software artefacts, which, unfortunately contemporary supercomputers still lack. We introduce vClusters (versatile software-defined clusters), which is based on Infrastructure-as-code (IaC) technology. vClusters approach is a unique fusion of HPC and cloud technologies resulting in a software-defined, multi-tenant cluster on a supercomputing ecosystem, that, together with software-defined storage, enable DevOps for complex, data-driven workflows like grid middleware, alongside a classic HPC platform. IaC has been a commonplace in cloud computing, however, it lacked adoption within multi-Petascale ecosystems due to concerns related to performance and interoperability with classic HPC data centres' ecosystems. We present an overview of the Swiss National Supercomputing Centre's flagship Alps ecosystem as an implementation target for vClusters for HPC and data-driven workflows. Alps is based on the Cray-HPE Shasta EX supercomputing platform that includes an IaC compliant, microservices architecture (MSA) management system, which we leverage for demonstrating vClusters usage for our diverse operational workflows. We provide implementation details of two operational vClusters platforms: a classic HPC platform that is used predominantly by hundreds of users running thousands of large-scale numerical simulations batch jobs; and a widely used, data-intensive, Grid computing middleware platform used for CERN Worldwide LHC Computing Grid (WLCG) operations. The resulting solution showcases reuse and reduction of common configuration recipes across vCluster implementations, minimising operational change management overheads while introducing flexibility for managing artefacts for DevOps required by diverse workflows.

Keywords
HPC, cloud, infrastructure-as-code, co-design, devOps, grid middleware

Introduction

The demand for High-Performance Computing (HPC) and data analysis is growing across various communities, in line with the rapid digitalisation of scientific methods and tools. At the same time, historically distinct resource requirements for supercomputing and large-scale data analysis are now converging as overheads for data movement are increasing with growth in experimental and simulation data generation capabilities. However, vendors and operators of supercomputing technologies limit hardware and software configuration changes outside of the major 4–6-year lifecycle management timeframe to minimise service disruptions, particularly for 24 × 7 × 365 operational environments.

[1]University of Bristol, Bristol, UK
[2]Swiss National Supercomputing Centre, Lugano, Switzerland
[3]ETH Zurich, Swiss National Supercomputing Centre, Lugano, Switzerland

Corresponding author:
Sadaf R Alam, University of Bristol, Great George Street, Bristol BS8 1QU, UK.
Email: alamcscs@gmail.com

Emerging machine learning and data science frameworks configuration and deployment pipelines leverage cloud technologies for abstractions and virtualization, creating a convergence from a software development and deployment perspective. This convergence enables efficient and secure software development and deployment using continuous integration, delivery and deployment (CI/CD) pipelines, from edge instruments to on-premise and off-premises IT resources to supercomputers. DevOps, defined as a closed-loop delivery and operational cycle for IT services, facilitates readily available new features and fixes through highly efficient development, testing, staging and delivery to production pipelines for developers and operators.

To achieve this, standard technologies such as containers, orchestration engines, microservices architecture (MSA), RESTful APIs, secure identity and access management (IAM) protocols, and Infrastructure-as-Code (IaC) are employed, while maintaining separation of concerns and enabling secure, common abstractions for independent development, testing and validation through standard interfaces and protocols. Equally essential are widespread availability of common tool chains for packaging, integrating, distributing and running software (Artifactory) (Docker) (GitHub) (Open Container Initiative).

The supercomputing infrastructure however largely remains outside of this realm of mainstream CI/CD pipelines due to primarily four constraints, which subsequently limit DevOps within a supercomputing ecosystem:

1. Bespoke software development and deployment environments because HPC sites provide vendor and site-specific programming and runtime environments. Configurations are not fully reproducible beyond the installation site.
2. Bespoke, tightly integrated, system software stack. For instance, Slurm configuration is either optimised for HPC or HTC (high throughput computing) payloads.
3. Bespoke change management, with custom version control and limited rollback options that is defined by the site as a global operation impacting multiple layers of the software stack.
4. Bespoke, air-gap systems in terms of Unix style access control roles and SSH access methods, which are site-specific limiting options for running distributed CI/CD pipelines securely on a shared environment.

Our proposed vCluster (versatile software-defined clusters) on supercomputers approach addresses the first three, while facilitating the fourth one, which has been listed for completeness. Different public cloud providers (AWS ParallelCluster) (Azure CycleCloud) (GCP Slurm Cluster) and independent providers such as cluster-in-the-cloud (Cluster-in-the-Cloud) have developed similar solutions using standard abstraction such as TerraForm. An open-source, abstractions-enabling infrastructure programmability is not available for a supercomputing ecosystem like Cray-EX Shasta, the supercomputing platform for November 2022 number 1-ORNL Frontier (Frontier Supercomputer) and number 3-CSC LUMI (LUMI Supercomputer) list of fastest computers (Top500).

Table1 highlights typical workflows for each constraint as motivating examples, the requirements and related mainstream CI/CD approach. The proposed vCluster solution in turn fuses the standard approach with supercomputing characteristics such that the resulting solution retains the defining characteristics of both supercomputing and cloud technologies.

With decades of experience of navigating these constraints at CSCS, we have been co-designing and co-deploying a set of solutions, inspired by the cloud technologies for deploying software, platform and infrastructure services (X-as-a-Service model) since 2015. These virtualisation and abstraction technologies include container runtime for HPC, RESTful APIs for HPC to address the fourth constraint of the list, and recently, IaC for HPC (vClusters) for supercomputing environments. vCluster is a unique fusion of HPC and cloud technologies resulting in a software-defined, multi-tenant cluster on a supercomputing ecosystem, that, together with software-defined storage, enable DevOps for the entire grid middleware stack alongside classic HPC platforms. With respect to the classic configuration where an HPC cluster can serve different usage patterns through a batch resource scheduler for supporting build, testing, interactive and long-running simulations, the versatility in vCluster enables creation of separate tenants, and simplify cluster configurations. For instance, an application-build cluster is a different tenant from a production batch cluster, using the shared hardware infrastructure while reusing common configuration recipes. Likewise, a cloud-native container orchestration cluster can be configured alongside a slurm cluster, as two separate tenants. Two major advantages of using a shared, HPC networked infrastructure with multiple vCluster tenants is the flexibility in sizing or elasticity of individual clusters as per quality of service (QoS) such as high availability and enabling isolation of one cluster tenant from another.

With the vCluster solution, we maintain a look-and-feel of classic HPC workflows such that the changes are transparent to users, for instance, how they access an HPC cluster, how jobs are submitted and how data is staged. Solution design and development follow a collaborative co-design approach with domain experts and vendors; therefore, these are highly impactful specifically for 24 × 7 × 365 HPC services, and for quick uptake and usage by targeted communities. A further advantage that is typically not available for large-scale supercomputing ecosystems is

Table 1. Key constraints for realising users and platform-driven CI/CD pipelines on contemporary supercomputing ecosystems. Typical HPC configurations are listed as examples highlighting the reasons why CI/CD pipelines and tooling that are commonplace in software development teams cannot be easily leveraged in a classic supercomputing ecosystem. These configurations are compared with the requirements for CI/CD pipelines that are often a prerequisite for DevOps, enabling close-loop development, testing, delivery and deployment of software artefacts. Standard approaches are listed for addressing these requirements thereby enabling DevOps opportunities across the supercomputing ecosystems, specifically with proprietary hardware and software stacks such as the Cray-HPE Shasta supercomputer.

Constraint	Typical HPC configuration	CI/CD requirements	Standard approach
1	Some sites provide test and development systems (TDS) on separate, significantly smaller-scale clusters for building and testing before changes are applied to production cluster. Production clusters are taken offline for environment updates	Reproducible CI/CD pipelines including automated testing and deployment, and user-driven change management are needed, independent from the cluster infrastructure operator, where target cluster configuration can be reproduced autonomously	Reusable and customisable programming environment and runtime configurations representing the relevant state of a production cluster that can be instantiated independently as multi-tenant, software-defined infrastructure (machines or clusters)
2	Platform users are often restricted to site-specific images, file systems (for instance parallel file system for HPC) and site-specific slurm configuration which are either optimised for HPC or HTC workloads	Platform developers in particular need an ability to compose and customise system stack artefacts such as slurm and file system integration before operational deployments	Reusable and customisable configuration images per cluster tenant that can be instantiated independently by platform developers as isolated tenants, for their CI/CD pipelines
3	Supercomputing vendors and sites have single, global configurations with predefined change management frequencies for compute, storage and networking software updates	Infrastructure configuration is managed like any software, or IaC approach where each change result in a new unique configuration or state, with rollback as a build-in option	Decoupled infrastructure and platform provisioning using DevOps approach. CI/CD pipelines of different artefacts are independently instantiated and validated by reusing configuration recipes

availability of smaller (in size), on-demand and ephemeral vClusters tenants for testing, developing and staging software before changes are rolled out to the production tenant.

The flagship Swiss National Supercomputing Centre's Alps ecosystem demonstrates the design principle of vClusters for two characteristically different workflows. The Alps supercomputing ecosystem has a highly heterogeneous infrastructure to fulfil different QoS constraints from diverse customers, one HPC and another data-driven HTC. Therefore, the vCluster approach is essential in order to manage versatile configurations using a software-defined infrastructure management philosophy such as IaC to address the constraints identified in Table 1. Alps is based on the Cray-HPE Shasta EX supercomputing platform. The microservices-based architecture of the Shasta management system is leveraged for IaC for HPC. We provide implementation details of two operational clusters. One is a classic HPC platform that is used for large-scale numerical simulations. Another is a widely used, Grid computing middleware platform used for data-driven, HTC computation for the CERN Worldwide LHC Computing Grid (WLCG).

Our main contributions are:

- extension of IaC concept for supercomputing called vCluster on Cray-HPE Shasta supercomputer software stack to align with software-defined, cloud-native compute and storage management systems. This addresses three main constraints of the bespoke supercomputing ecosystems by enabling programmable instantiation of multi-tenant build, test, stage and operational vClusters
- demonstration of full coverage of a data-driven HTC workflow leveraging HPC and cloud interoperability as well as a classic HPC cluster within the Alps ecosystem reusing and customising multi-tenant vClusters recipes
- deployment of CI/CD pipelines using standard tools, configuration recipes and interfaces maintaining separation of concerns between different teams, for example, Grid software development teams and site operational teams reusing vCluster recipes for build, test, stage and operational environments

The article outline is as follows: the next section provides details of the vCluster solution with respect to contemporary approaches of deploying and operating HPC clusters. Implementation details of vCluster in Cray-HPE Shasta EX microservices-based architecture in the following section. The two sections after Shasta outline implementation details the classic HPC platform and coverage for the WLCG Grid middleware platform. This is followed by an evaluation and analysis of advantages of the vCluster approach, which is followed by an overview of the related work. We then conclude our study and provide an outlook for the future directions.

vClusters – versatile software-defined HPC and cloud clusters

vClusters (versatile software-defined clusters) are built on Infrastructure-as-Code (IaC) with an aim at making supercomputing ecosystems mainstream for developers of artefacts deploying application software and domain platforms using contemporary CI/CD pipelines, which can be instantiated on any infrastructure, independently by application and platform developers. IaC is an approach for provisioning and managing hardware resources such as computing, storage and network in a programmable, automated manner, much like a versioned-controlled, software ecosystem (Morris 2016). vClusters extends the IaC concept with HPC-specific criteria of close-to-metal performance and scaling efficiencies. IaC approach for provisioning and management relies on key principles of software engineering, namely, configuration management, version control and automation. This is due to the growing needs for faster and robust delivery of new services by supporting multiple continuous integration and delivery (CI/CD) pipelines managed by diverse teams, from web applications to infrastructure operators, using what is called DevOps (automated development-operations) approach. This is essential because scientific workflows increasingly leverage not only HPC methods but also data science techniques such as Machine Learning (ML) and Artificial Intelligence (AI), often relying on distributed, on-premise and off-premise (cloud) resources. High Throughput Computing (HTC) is another usage model that does not map well with classic supercomputing style configuration and resource management systems.

Typically, IaC concepts are supported through virtualisation of the infrastructure resources and containerisation of applications, decoupling hardware from the functional elements. There lies the challenge for HPC and supercomputing where virtualisation of hardware and containerisation of software stacks are uncommon due to concerns about performance. Fusion of HPC and cloud technologies enable extensibility of IaC for supercomputing, which we will demonstrate in the article. Two models exist for IaC, namely, imperative approach, and declarative approach (HPE IaC). A declarative approach defines the desired state of the system and the current state of different objects, which are configured and maintained by the IaC tool chains. In contrast, an imperative approach lists specific commands needed to achieve the desired configuration, in a specific order of execution. We exploit both approaches for IaC as the technology is evolving for HPC stacks and their efficacy vary depending on the desired objectives and the level of abstraction needed for the configuration management.

Figure 1 shows the setup of a classic HPC cluster and composable vCluster configurations, using a high-level classification of the resource management plane, the infrastructure, platform and user level software layers. A classic single-tenant HPC cluster has a tightly integrated environment from the infrastructure management plane to the programming environment layers. Although the unique user level software can be installed, often there are dependencies on MPI and driver versions that are part of the tightly integrated stack, often managed by system administrators who are responsible for the resource plane. Similarly, a single resource management and scheduling system, such as a batch scheduling system is configured to support different usage modes such as debugging, long running jobs, and interactive jobs. The versatility in vCluster is to create separate tenants and simplify cluster configurations, for multiple usage scenarios such as developing and testing a software, checking readiness for production, and operational deployment. Moreover, each layer can be configured separately and is independently managed. This gives flexibility for independent software development teams, for instance, numerical libraries developers, application software teams, and as well as site operators to deploy CI/CD pipelines in an automated manner assuming standard abstractions are being used.

In order to fully support the vCluster model, we rely on an immutable infrastructure approach. Immutability in IaC essentially means that once a configuration is generated for an image it is versioned and is never updated in place. A new version must be generated for even minor changes and patches. The idea is that there is always a state that can be restored. There are trade-offs with this approach, specifically maintaining a large number of states of machines. The advantages include autonomous and reproducible CI/CD pipelines by diverse teams. Typically, large-scale classic supercomputing installations follow a mutable model due to the complexities associated with upgrading the entire, vertically integrated stack. Different elements can be updated in a compute node image for instance, drivers, MPI libraries and security patches. Often this leads to untraceable and tedious debugging and troubleshooting on supercomputing platforms when regressions and errors are reported because it is impossible to go back to the previous known state of the system since it is not kept.

The resource plane is isolated and is separately managed from any of the vClusters. In fact, one way to view the distinction is that the system administration team is focused on managing the health of the resource plane components, and is responsible for change management and troubleshooting of resources independently of any of the top, X-as-a-Service (XaaS) layers. Each XaaS layer can be managed by different teams (roles), which do not have any privileged access to the resource plane. Thus, a role-based access control (RBAC) is a key requirement for implementing

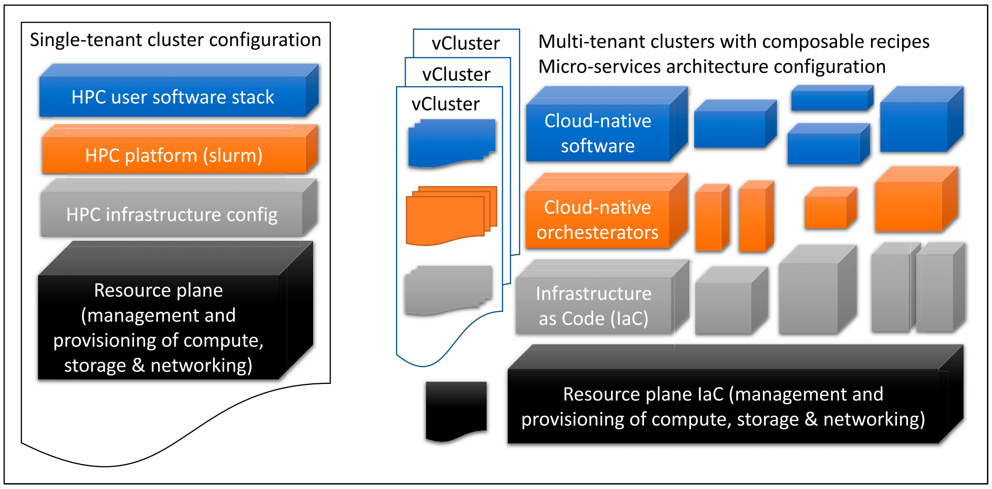
Single-tenant cluster configuration
HPC user software stack
HPC platform (slurm)
HPC infrastructure config
Resource plane (management and provisioning of compute, storage & networking)
vCluster
vCluster
vCluster
Multi-tenant clusters with composable recipes
Micro-services architecture configuration
Cloud-native software
Cloud-native orchesterators
Infrastructure as Code (IaC)
Resource plane IaC (management and provisioning of compute, storage & networking)

Figure 1. Comparison of two distinct approaches for provisioning, deploying, operating and change management of a supercomputing system. On the left, there is a classic approach where there is a single tenant and single configuration for HPC infrastructure, platform components such as resource management and job scheduling system and user stack including compilers, MPI libraries, programming tools, etc. Although variants of the user software stack are possible through module files, typically there are strict dependencies on lower stacks such as MPI, drivers and the operating system. System administrators and engineers are responsible for maintaining and upgrading the entire, vertically integrated stack. Therefore, the change frequencies and customisation options are often restricted, and several operations are not allowed as a risk mitigation due to the lack of separation of concerns. These operations often require high level of privileges (admin). On the right, we have the vClusters approach where a cluster configuration is defined as a recipe comprising configuration of artefacts from user, platform and infrastructure levels for maintaining separation of concerns. The resourcing layer is separate and is managed by the data centre system administration team where the main task is to maintain the health of underlying resources. vClusters therefore rely on maintaining the separation of concerns by defining clear interfaces between the layers for composibility such that independent CI/CD pipelines of individual software components can be managed. Note that a vCluster can run HPC resource management software like slurm. Since slurm does not follow a microservices architecture (MSA) it needs to be integrated such that it can be composed alongside other microservices. Cloud-native terminology is used to specify technologies that are MSA compliant and therefore fulfil DevOps requirements.

vClusters. RBAC is commonplace in the identity and access management (IAM) systems of cloud-native technologies while classic HPC uses a typical Unix style access model. IAM and change management of the resource plane and vCluster layers are distinct. Another characteristic feature of cloud-native solutions is MSA compliance where each microservice can be managed and updated independently. All microservices operate with a common IAM and security model based on RBAC and policies. Depending on the role, a microservice from each layer can be configured to obtain a platform service configuration such as an HPC platform or a canonical slurm cluster, or a complex data-driven middleware of the Grid platform, which itself is composed of multiple services including an HTC cluster. These details will be provided in the next sections.

The key benefit of an IaC vCluster approach on Shasta supercomputer for our study is maintaining separation of concerns for the development and deployment of non-cloud-native HPC software stacks on an infrastructure that follows an MSA architecture. Our design target is to enable development-operations (DevOps) style change management and to maximise productivity of diverse teams' ability to configure their continuous integration and delivery (CI/CD) pipelines. As a result, both platforms demonstrated in this article have smaller (in size) and ephemeral vClusters tenants for testing, developing and staging software before changes are rolled out to the production tenant. Conversely, the recipes can be potentially adopted and deployed on the non-Shasta ecosystems provided similar IaC abstractions are available.

Cray-HPE Shasta EX system and software stack

The HPE-Cray EX system and software management stacks support IaC capabilities through key innovative technologies (Cray System Management). These systems are characterised by two distinguishing features that are relevant for our study:

- Software-defined, Ethernet-compliant, high-performance networking stack – interoperability with the data centre networks has been among the key designed objectives of the Cray-HPE Slingshot interconnect technology, in addition to support for the HPC, MPI payloads (Khorassani et al., 2022). As a result, it is compatible with Ethernet while offering additional essential features for the low-latency and high-bandwidth needs of HPC applications such as adaptive routing and congestion control. A further enhancement has been the introduction of software-defined features for creating isolated, network tenants, similar to other contemporary cloud-native networking technologies. It provides system-wide service classes, to control and manage network bandwidth, among individual jobs ensuring high priority jobs get the network resources they require, in a programmable manner.

- Adoption of cloud-native architecture for HPC management – this has been done by separating administrator services and services for the developers or end users into different access control zones, namely, the management plane and the managed or compute plane. Instead of using physical dedicated servers for administrative services for a tightly coupled, vertically integrated orchestration for services like hardware provisioning, configuration, boot and system management, it follows MSA design principles. The full separation of the management plane from compute plane and the shift towards a service-based micro-architecture allows each plane to run and be upgraded independently without impacting the other. This significantly minimises service interruptions and down times for users. Furthermore, the interfaces are built with published APIs that allow users and system administrators to interact with components of the system in order to orchestrate privileged operations in a flexible manner.

Support for network virtualisation for HPC communication protocols, and, critically, MSA for the system management are key enablers for DevOps. For instance, the new stack permits the creation of multiple isolated groups or tenants (compute and networking) which in turn can be configured as multiple isolated and elastic batch clusters with Slurm or PBS, or Kubernetes clusters in a programmable manner (Jette et al., 2003) (Hightower et al., 2017) (Zhou et al., 2021). Availability of such programmable partitioning without virtualisation overheads has been the main motivating factor for implementing vClusters on Shasta, from a configuration management point of view. The compute nodes are not shared or virtualised (not running a hypervisor) and instead are provisioned bare-metal.

Additional features of Cray-HPE Shasta EX systems include the support of highly heterogeneous compute nodes. The first Exaflops supercomputer, ORNL (Frontier Supercomputer), has AMD CPU and GPU nodes. Similar configuration is available for the largest partition of the LUMI supercomputer at CSC, Finland (LUMI Supercomputer). LUMI supercomputer also has multi-core only nodes with dual socket AMD GPUs. Another Cray-HPE Shasta EX supercomputing platform at NERSC, LBNL, has Nvidia GPU accelerators. Planned systems include nodes with Nvidia ARM processors, and next generation of AMD, Intel and Nvidia GPUs. We believe a vCluster approach will be highly effective for managing complex user, platform and system software stacks on highly diverse, heterogeneous supercomputers, including Alps.

The first phase of installation of Cray-HPE EX hardware with the Shasta software stack was completed during the fourth quarter of 2020 at CSCS. A production Slurm HPC virtual cluster instance was made available to external users during the first quarter of 2021 (CSCS Eiger Cluster). Since then, we have been gradually co-designing and evaluating the environment to achieve our multi-tenant vClusters design objectives, as the Shasta hardware and software stacks continue to evolve. The HPC cluster services remain available and used by users for submitting their MPI jobs while the management stack is going through major updates and configuration changes. This can be attributed to the adoption of multi-tenant vClusters approach on Alps that is detailed in the next section. While the work is in progress for fully developing the software-defined networking and the IaC management stack of clusters and machines, we have exploited a subset of existing features of Cray-HPE Shasta to create multi-tenant HPC vClusters for users that have been in operation since 2021.

HPC Platform vClusters on Alps

Historically, the design objectives of system and software management stacks of a supercomputer do not include requirements for programmable access from other on-premise and off-premises resources, or applications. This is because the objective for an HPC cluster is to primarily run large-scale MPI jobs, through a batch scheduling system. Users of such a system define number of nodes or resources and duration of a job, and then wait until the requested configuration becomes available within an HPC cluster that is shared by multiple users and their jobs. These systems typically have two operating zones, a user zone to compile and run applications, and an admin zone, for everything else, such as configuration management, provisioning, and orchestration of system and software stacks. Several change management operations are not allowed because there is only one tenant; therefore, any configuration change can potentially impact the entire supercomputing infrastructure (deemed too high risk for an expensive resource). Hence, users use the supercomputing system through batch jobs (running applications compiled on the system) and system administrators try keeping the ecosystem healthy by taking periodic down times for configuration changes, and for applying patches and fixes (Allen et al., 2020) (Lundin and Fisher 2019).

Figure 2 shows the procedure for configuring a vCluster within the Alps ecosystem and instantiating it such that

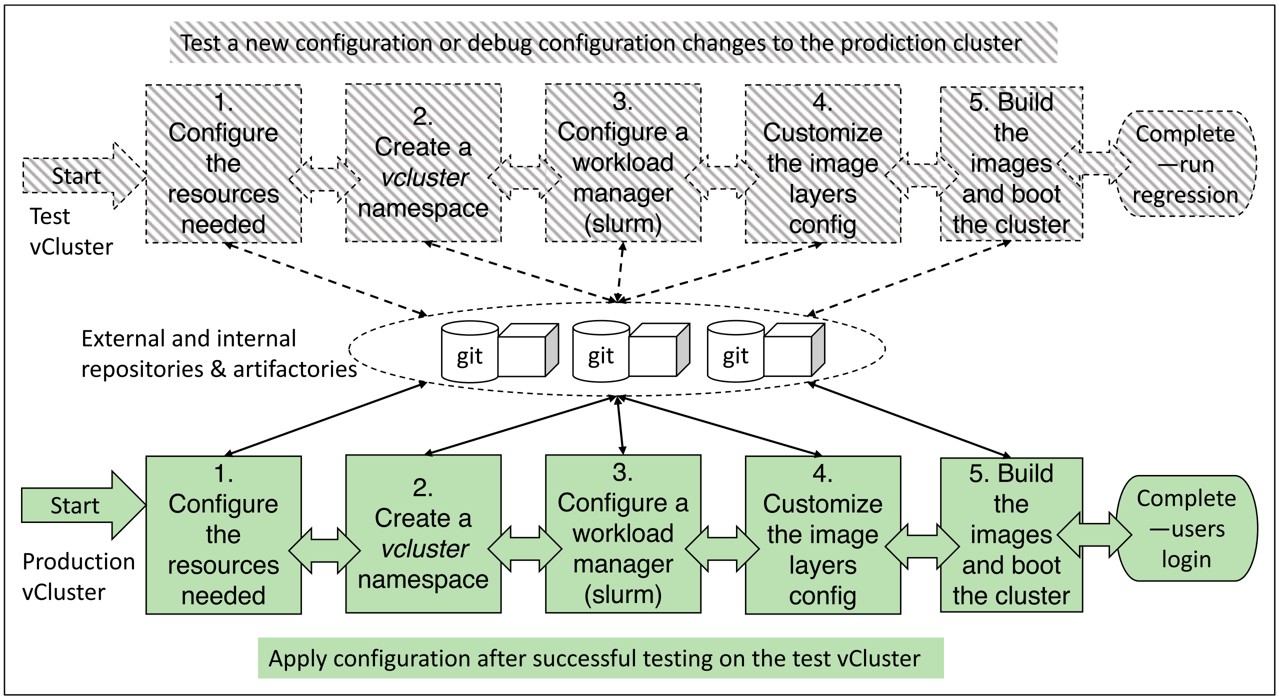
Test a new configuration or debug configuration changes to the prodiction cluster
Start
Test vCluster
1. Configure the resources needed
2. Create a vcluster namespace
3. Configure a workload manager (slurm)
4. Customize the image layers config
5. Build the images and boot the cluster
Complete —run regression
External and internal repositories & artifactories
git
git
git
Start
Production vCluster
1. Configure the resources needed
2. Create a vcluster namespace
3. Configure a workload manager (slurm)
4. Customize the image layers config
5. Build the images and boot the cluster
Complete —users login
Apply configuration after successful testing on the test vCluster

Figure 2. Current procedure for setting up vClusters on Alps Cray-HPE Shasta supercomputer. Two instantiations are shown, one for a test HPC platform and another for an operational HPC platform. The test vCluster can be on-demand and the configuration and the cluster can be deleted, and resources can be freed and added to the central resource pool. The instantiation of a test cluster can be triggered by different needs. There could be a new version of software or security patch that needs to be applied to the operational platform, but first it must be tested and validated for operational integrity. Common repositories are used for existing configurations and templates. Any modification of a template is updated into the repository to maintain immutability. Versions are maintained and updated using common tools such as Github and Artifactory with roles and access controls for running different CI/CD pipelines. There are both internal and external such as vendor provided software. This is particularly important for deployment testing on the operational cluster. Each step is proceeded after the next such that issues can be troubleshooted and an update can be rolled back. Note that the production vCluster HPC platform need to fulfil the service level agreements (SLAs) for health, reliability and availability metrics for the customers.

users can perform typical operations seamlessly such as login to the cluster, compiling an application and submitting a job. Two cluster tenants are shown in the figure, a test and development (TDS) environment that can be on-demand and ephemeral and a production HPC cluster. Major parts of the configuration recipes are reused for build, test, stage and production HPC platforms instantiated using vClusters.

There are five main steps of the procedure that each recipe follows to generate vClusters on Alps supercomputers. Within each step, different configuration options can be selected from the repository, or a customised one can be built. A couple of steps require privilege roles to control access to the resource or management plane, for instance, requesting different numbers and types of nodes from the resource pool. The objective is to eliminate privilege access and make these available using APIs that enforce roles and policies-based access controls. Policies and QoS govern if a resource request can be granted. The main steps are:

1. Request and configure the resources
2. Create a vcluster namespace on the resource plane
3. Configure a workload manager
4. Customise the image (layers configuration)
5. Build the images and boot the cluster

The vCluster layers shown in Figure 1 are largely mapped onto these five stages to distinguish our approach from a single-tenant HPC cluster configuration. First, the resourcing layer is isolated and is managed separately through a role and policy-based access control. The first step assumes that this layer is configured already. Second, the IaC configuration is the starting point for the procedure, as step 1, which is followed by step 2 for interfacing a new tenant registration with the resourcing plane (for inventory). Step 3 represents the orchestration layer, which in the case of an HPC cluster configures a batch workload manager (slurm). Step 4 involves creating an image with several options that are typically not available in a classic provisioning approach. This step is customisable, giving configuration opportunities for setting up a full programming and runtime stack, or leaving these configuration options to users to bring-their-own. All configurations are kept in standard repositories and artifactories (CSCS uses JFrog) for managing versions and access controls to the configuration files that are available for sharing and creating custom branches for the development and sharing of different software artefacts.

In the example below we assume the name of the vCluster is cscs-test-hpc.

Request and configure resources

In this step, the resources (physical nodes) are allocated to a vCluster. For our cscs-test-hpc cluster example, we allocate 50 CPU only (multi-core AMD) nodes, from the Alps resource pool. The first step is to clone a repository for the template management and create a folder for the vcluster, naming compute nodes (cn) and login or user access nodes (uan). The cscs-test-hpc cluster will include all cscs-test-hpc-cn and cscs-test-hpc-uan nodes.

The next task is to move available nodes needed from the resource pool to template-management/hsm/free_[cn—uan] on the respective vcluster[_cn,_uan] and copy all the nodes on the general vcluster folder. At this stage, if no free resources are available such as uan, some nodes can be repurposed. The repository for the vcluster is then committed and pushed to the general repository. The updates can be checked whether the requested configuration throws any errors or exceptions before proceeding to the next stage.

Create a vcluster namespace on resource plane Kubernetes

Kubernetes namespaces are a way to organise clusters into virtual sub-clusters. From the management node, the namespace is created. This is also a way to manage, organise, reuse and track all vClusters on the resource plane.

Configure a workload manager (Slurm for HPC platform)

For the cscs-test-hpc example, we use Slurm, one of the commonly used resource management and job scheduling systems for HPC clusters. Slurm is made-up of two main parts: Slurmctld job manager; and slurmdbd job accounting DB. Slurmctld will be a dedicated vCluster pod, but Slurmdbd can be configured to be local (a dedicated pod will be used) or remote or external resources can be used. Like the previous steps, existing templates will be reused. A DNS entry can be requested for the vCluster such as cscs-test-hpcctld.cscs.ch.

The next step is to create a slurm cluster using helm, which is a way to manage Kubernetes applications. Likewise slurmdb, it will be setup using the cscs-test-hpc vCluster name in the repository file. Then the authentication service for slurm, namely, Munge will be configured. Once these steps are configured successfully, the slurm cluster can be deployed.

Customise the image (layers configuration)

The images are customised with (Heap 2016) and re-packaged as a new image. The customisation is done through different layers grouped under CSCS git repositories. These include:

- CSCS-specific vCluster template configurations: this contains CSCS-specific customisation. Here relevant files are copied and can be adapted from an already working vCluster, based on different roles
- Cray Operating System (COS) configurations: file systems such as scratch, user and apps are added here.
- Login or user access node configurations: same as COS but in addition to file system mount, LDAP for the login nodes is configured here
- Cray Programming Environment (CPE) configuration: this is optional and performed if CPE is required. Bring your own programming environment option can be added later
- Configuration Framework Service (CFS) configuration: this defines the layers in the images to be installed on the compute (COS) and login (UAN) nodes

Build the images and boot the cluster

With the generated CFS config, we can submit the command to build the image. This will launch a build of the compute image which returns immediately because it launches a pod on the K8s management plane to build the image. This process can be monitored as needed. Upon successful completion, the details can be checked using the name provided during the build (cscs-test-hpc-csm-1.xyz in this example).

The image artefacts are stored in an internal Cray S3 bucket. All the information is then gathered for the image ID to create the boot orchestration service (BOS) configuration, using the same approach as CFS. Upon successful completion, the new BOS template is available where details can be checked using the name provided during the build (cscs-test-hpc-csm-1.xyz in this example). Alternatively, if different images are needed for the compute nodes, the details of the config can be obtained with cray BOS session template, which describes cscs-test-hpc-cos-templateXX (to check the various IDs, to manage the booting of individual nodes, a node can be from the list of compute nodes for cscs-test-hpc-cn group).

Similar steps are performed to boot an accelerator node which has additional components such as a GPU driver and a programming environment for accelerators.

Finally, a vCluster called cscs-test-hpc is booted. A user can login to cscs-test-hpc.cscs.ch if they are in the vCluser LDAP and are authorised to submit jobs. The configuration and instantiation details are transparent to the end users. Similarly, applications such as automated regression tests or a CI/CD pipelines can access the login node of the HPC cluster. If the programming environment is not part of the configuration, users or applications will bring in and deploy the environment as part of the build and testing pipelines.

WLCG Platform vClusters on alps

The Worldwide LHC Computing Grid (WLCG) provides and operates the Grid infrastructure used by the experiments of the Large Hadron Collider (LHC) at CERN for their data processing (WLCG). It is an instantiation of Grid Computing, which is defined as a collection of distributed resources to perform collective tasks (a global collaboration of around 170 computing centres in more than 40 countries, linking up national and international grid infrastructures). WLCG's mission is to provide global computing resources for the storage, distribution and analysis of the data generated by the LHC. Already CERN workflows have one of the unique storage and data transfer requirements, reaching close to 100 Petabytes per year. The significant amount of data to be distributed and analysed, the number of collaborating centres and users and the diversity of the underlying resources make the WLCG the largest and most complex research Grid currently in operation (CERN Tape Archive). A substantial growth in data generation, storage and processing is anticipated for the next generation of experiments called Run 3 (Fartoukh et al., 2021).

CSCS operates a Swiss Tier-2 in the WLCG for the Swiss Institute of Particle Physics (CHiPP), mainly to analyse data coming from the LHC experiments, namely, ATLAS, CMS, and LHCb at CERN. Compared to the vCluster configuration for an HPC cluster, the setup for the WLCG platform has significantly higher complexity and external dependencies for the middleware stack. Nevertheless, using the vCluster approach, we can extend the HPC cluster configuration to an extreme data-driven HTC cluster configuration, and customise recipes for user access, slurm controls and software-defined, multi-tenant, HTC-oriented file systems. Previous studies have demonstrated that the middleware for virtualised computing and storage stacks aren't interoperable with classic, Petascale supercomputing systems (De et al., 2021) (Boccali et al. 2021). The middleware has been partially containerised (Roy et al., 2020). Our vCluster approach maps the entire middleware using three compute and storage management solutions, namely, Slurm, Kubterntes and Ceph (on-premise software-defined storage for HTC-friendly data services) (Ceph).

Grid middleware

Grid middleware is composed of different components that have diverse requirements in order to fulfil the key objective of leveraging distributed resources to perform collective tasks. On the one hand, these require high throughput computing (HTC) jobs on batch-oriented HPC like clusters. On the other hand, there are a large number of shared, long running services that do not map on to a batch-oriented environment. Hence, the implementation of the full stack requires discrete, highly flexible infrastructures to support bespoke computing and storage configuration stacks. Implementation details of vClusters are transparent to the end users of the grid services.

WLCG comprises four layers

- Physics software: these include software tools specifically designed to satisfy the changing demands of the high energy physics community. ROOT, a set of object-oriented core libraries used by all the LHC experiments; POOL, a framework that provides storage for event data; and other software for modelling the generation, propagation and interactions of elementary particles. Historically, these software elements do not map onto supercomputers where computing nodes do not have access to the Internet.
- Middleware: Grid projects supply much of the software that manages distribution and access, as well as job submission, user authentication and authorisation. They also supply software known collectively as 'middleware', defined as essential software for enabling physics applications. In this article, we will demonstrate Advanced Resource Connector (ARC middleware) and dCache virtual file system (Dcache) using vClusters on Alps.
- Hardware: Each Grid centre manages a large collection of computers and storage systems. CSCS is classified as a Tier-2 centre and supports ATLAS, CMS and LHCb Virtual Organizations (VOs). A VO is defined as a group of individuals or institutions that agree on sharing resources according to defined policies and, in the case of WLCG, is usually tied to an LHC experiment with the same name. One of the requirements by the VOs is to regularly update the necessary software so that information is available to the overall Grid scheduling system, which decides which centres are available to run a particular job. The supercomputing ecosystems typically operate on a strict change management and upgrade schedule. Hence, both system and application virtualization are essential for co-locating Grid middleware on supercomputing infrastructure.
- Networking: file-transfer service has been tailored to support the special needs of grid computing,

including authentication and confidentiality features, reliability and fault tolerance, and third-party and partial-file transfer.

In order to support interoperability across hundreds of resources distributed over multiple countries, the WLCG software stack is centrally distributed and updated. This approach is in contrast with a highly customised, vertically integrated supercomputing ecosystem where the operating system, the application software, monitoring, access controls and change management follow site-specific requirements and processes. With the new capabilities that Alps software-defined infrastructure ecosystem brings, we were able to take advantage of the vCluster and software-defined infrastructure cloud technologies while meeting abstractions and constraints for HTC and data-driven middleware, using programmability and composibility of IaC approach for WLCG platform recipes.

Middleware configuration

Grid middleware is composed of different elements that, in the case of a regional Tier-2 like CSCS, can be summarised at a high level as three main sets of service abstractions:

1. Compute Element (CE) – ARC
2. Storage Element (SE) – dCache
3. Shared services

These components are utilised using custom configurations by the VOs corresponding to the different experiments working with the LHC. The Tier-2 at CSCS supports the ATLAS, CMS and LHCb VOs, plus two additional ones used for testing (OPS and DTEAM). Typically the CE represents a submission gateway and the compute nodes or Worker Nodes (WN) in WLCG terminology, and a Local Resource Management System (LRMS) used to run jobs. There can be one or more such configuration per site, usually targeting unique hardware characteristics.

Unlike CE configuration per experiment, there is typically only one SE per site, often dCache. Nevertheless, each VO has a different way of accessing and utilising the SE. For instance, ATLAS uses ARC to stage in/out data from/to the local SE, and to make the data available on the job's runtime directory when the job starts. CMS and LHCb usually operate on streams of data connected to the SE that are managed while the job is running. SE details are provided in the data services section.

The design objectives as well as constraints (supercomputing ecosystem and WLCG middleware) are met by the following guiding principles for migrating Grid middleware services to Alps – vClusters and supported IaC ecosystems:

1. Remove CSCS-specific dependencies of all middleware and services by decoupling them from the infrastructure layer and porting everything to Kubernetes (and contribute and upstream our developments back to the community);
2. Migrate all computing resources to Alps. The resulting environment should be modular and versatile to accommodate other workflows with similar needs or that can leverage cloud-native approach;
3. Co-design and iterate in an agile manner, leveraging on collaboration with our colleagues from CHiPP to enable full coverage of WLCG middleware platform on a supercomputing ecosystem, maintaining separation of concerns of diverse teams.

Data services

Grid computing data services requirements do not map onto HPC file system services, particularly the concept of high-bandwidth low-latency POSIX file systems that is shared among all users of a supercomputer. There are two main challenges that we address by leveraging a software-defined storage solution for implementing dCache services. The first challenge is supporting high IOPS and meta-data operations on an HPC file system that are typically associated with HTC workloads. The design point of HPC file systems is to support scalable jobs with large file or block sizes while HTC workloads are composed of several small jobs and large number of small files. The second challenge is maintaining an optimal, global configuration recipe for all users and QOS per cluster or workflow requirement. Therefore, in the architecture of Alps, a dedicated or shared scratch file system has not been envisioned. Instead, the idea is to use Ceph to make jobs to have a 'local' disk space for CVMFS (CernVM-File System) and as scratch space that is then used by the jobs as local scratch. Our storage element (dCache) is implemented using using an open-source, software-defined storage system (Ceph).

As per our guiding principles, in order to improve the scalability of our dCache instance and benefit of the reliability that kubernetes provides, we implemented a kubernetes-based dCache by developing a Helm Chart. Helm is a package management system for Kubernetes-based deployments. This allows a flexible configuration in order to simply and quickly deploy multiple dCache instances with different configuration recipes for distinct use cases such as test environments and multiple customers. The solution uses Ceph as backend to store data, which is managed by the Ceph CSI (Container Storage Interface) components deployed on the Kubernetes cluster. Each dCache pool (which is a Stateful Set with replica one in kubernetes pod) is bound to a Kubernetes PVC (Persistent Volume Claim). The concept is to have the whole configuration in a code repository, which describes the dCache instance and is simple to redeploy if needed. Every change on the code is versioned to keep track of the development and of changes in the configuration recipes.

DevOps on alps

One of the key advantages of vClusters that we consider is enabling several, autonomous CI/CD pipelines and enable DevOps approach to the delivery of services. This is particularly important for distributed frameworks such as WLCG middleware that aren't designed specifically for HPC ecosystems. Figure 3 highlights the mapping of service description and orchestration on a vCluster (VC), which is essentially a configuration recipe that instantiates an HPC or Kubernetes cluster on Alps. Key terminologies used in Figure 3 are defined as follows. The different zones (DEV-development, Stage-STAGE, Production-PROD, Services) are used to fulfil the QoS and CI/CD pipeline requirements. This platform deals with three main types of services with different lifecycle and availability requirements as per service level agreements (SLAs) with the customer. To fulfil SLAs while enabling DevOps in a shared supercomputing ecosystem, we specify three different zones:

- We distinguish central or shared components that are necessary to make the site work and that behave like cattle (IaC Cattle Pet Analogy) (i.e., a restart of any of them isn't critical to the operation of the site, in WLCG terms). Several middleware components such as CVMFS or BDIIs fit this role: they rarely change, the impact of a single pod/service failure is not critical, and they mostly don't need certificates linked to specific IPs, nor use odd protocols such as xrootd or gridftp. This is the 'Services' zone.
- The 'DEV' zone is not persistent and is instantiated on demand for development purposes, but the 'STAG', 'PROD' and 'Services' zones are persistent. Both STAGE and PROD are under 'production' SLAs from a WLCG customer's or VOs perspective. Their configurations serve for testing changes with operational workflows of customers, as well as to enable the gradual introduction of new nodes/configurations (elasticity). One could argue that a development zone is not needed,

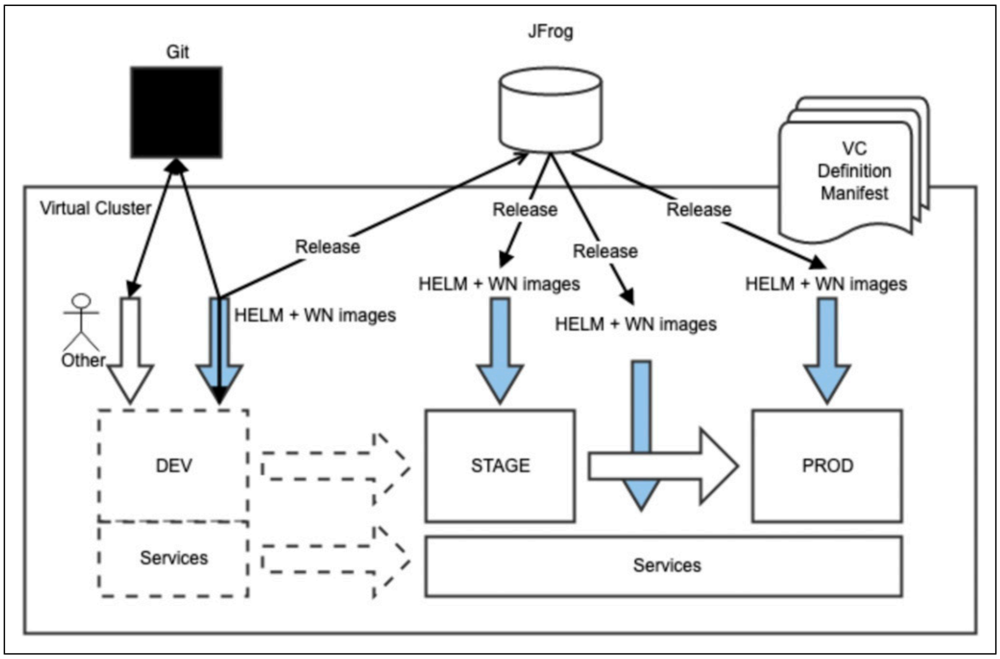

Figure 3. vClusters on Alps and on-premise software-defined infrastructure for mapping services for grid workflows on Alps. The implementation of the vCluster (VC) is divided into four zones to allow independent development and operation (IaC concepts of pet and cattle for named and unnamed clusters): Dev (cattle), Stage (pet), Production (pet) and Services (pet). DEV vClusters are instantiated on-demand and is ephemeral with the corresponding configuration recipes for the WLCG middleware services. Stage and production vClusters share production services. As the name suggests, after testing on the DEV vClusters, changes are staged with common releases such that part of the workload can start using the new configuration e.g. worker node (WN) images. All changes are transparent to the end users. The nodes can be gradually added from PROD to STAGE such that all experiments can validate and migrate their workflows. In case of regression or issues, changes can be rolled back on the stage and the configuration is discarded. Workarounds, patches and updates are first tested on the DEV environment. Components (composed of HELM charts and compute node images) include multiple middleware service configuration such as CVMFS Proxy, IAM, CMS Proxy, Dashboard, ARC, dCache, Slurmctld, Worker Node (WN), which can be tested in the DEV vCluster platform before migrating to STAGE. Standard tools like Github and JFrog (an artefact repository) are used at CSCS for managing CI/CD pipelines. Note that the PROD and STAGE are under the service level agreements (SLAs) from customers meaning that operational metrics for health, reliability and availability of production services and resources are being monitored for compliance.

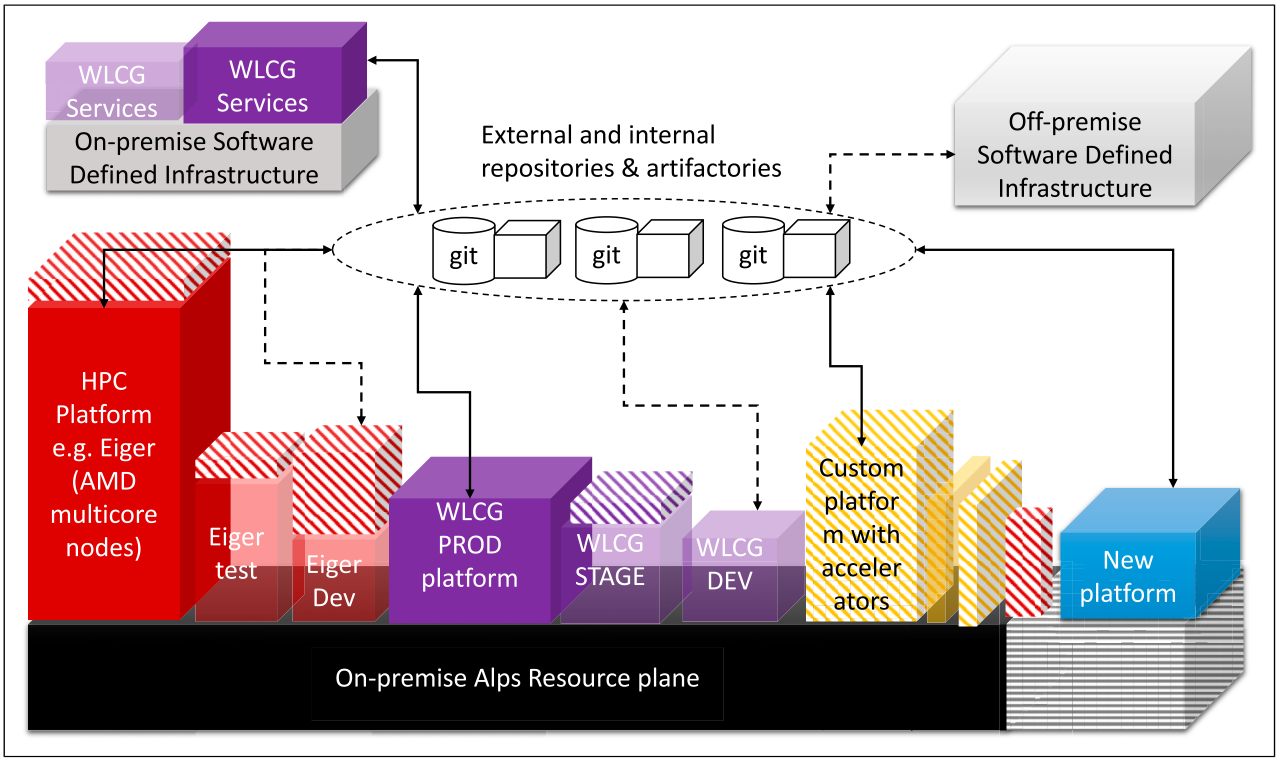
WLCG Services
WLCG Services
On-premise Software Defined Infrastructure
External and internal repositories & artifactories
Off-premise Software Defined Infrastructure
git
git
git
HPC Platform e.g. Eiger (AMD multicore nodes)
Eiger test
Eiger Dev
WLCG PROD platform
WLCG STAGE
WLCG DEV
Custom platfor m with acceler ators
New platform
On-premise Alps Resource plane

Figure 4. Alps supercomputing infrastructure is currently resourcing different platforms in operation including two use cases that have been explained in the article for the vCluster implementation: HPC and WLCG. The striped area on the resource plane shows existing and upcoming changes. Heterogeneous infrastructure resources continue to be added to Alps, within the same SlingShot-based high-performance network, which is partitioned in software for multi-tenancy. In addition, both on-premises and off-premises resources can be included for creating, reusing and deploying configuration recipes using common repositories and artifactories. The operational cluster is shown as solid boxes that have the required number of resources as per contract and SLAs. The patterned coloured boxes show possibility of elasticity for vClusters (adding or removing nodes as needed) while the shaded boxes show CI/CD staging ephemeral vClusters that can be instantiated on-demand. Additional platforms can be generated by reusing majority of existing configuration steps and repositories, for instance, custom platforms. Example of on-premise, non-Alps resources include Ceph software-defined storage for the WLCG platform's HTC file system configuration needs. The resourcing details are transparent to the end users. Currently, off-premises resources have not been deployed for any operational clusters.

but in this use case, many components cannot be tested locally on laptops because they have specific requirements that can't be satisfied with local development (i.e., grid certificates and compute node images).

- The 'PROD' zone contains components that need to be registered and stable for operation (computing, storage, ARC, etc.). dCache and ARC gateways (plus the computing resources behind them) fit this description as they need to be registered in WLCG (and with each VO) where SLA metrics are monitored and reported. Any failure or regression is considered critical in establishing the health of a site in the grid central ecosystem. Further technical implementation details and operational results are provided in (Gila et al., 2023).

Evaluation and discussion

We evaluate the design and implementation vClusters for the two distinct use cases that were presented as a demonstration to address three constraints:

1. Bespoke software development and deployment environments
2. Bespoke, tightly integrated, system software stack
3. Bespoke global, change management, with custom version control and limited rollback

At the time of writing this article, both platforms are in production and are operating with SLAs that predate Alps vCluster design approach. Figure 4 illustrates implementation of two platform use cases, additional platforms that are being developed using multi-tenant vCLusters on Alps, on-premise software-defined infrastructure and possibility of extending to remote IaC capable software-defined resources such as public cloud providers. Concretely, the configuration of all layers is managed using commonly used CI/CD pipelines and tools. Using standard technologies, we observed an increased reuse of recipes across platforms. For instance, historically, an application software developer will be dependent on the system administrator and will be constrained by the global configuration of the HPC platform on a supercomputer. The approach had some drawbacks, for instance, the system administration team was deemed responsible for ensuring health of all layers of a complex software ecosystem for diverse configuration needs. By separating concerns of different roles and layers of the deployment process in the vCluster procedure and enabling on-demand configurations for autonomously composing layers (test, stage and production tenants), testing and deployment can be automated without compromising overall health of the production platforms. Furthermore, changes can be rolled back within the layer minimising overhead for managing and validating changes for a vertically integrated, complex HPC cluster configuration.

At CSCS, the vCluster approach triggered two rather interesting phenomena. The first one being a convergence of common CI/CD tooling needs across teams, from scientific application software developers, platform engineers to system operators. There is an increased transparency as a result where previously there was a black box thinking for updates and change management. The roles and policy-based IAM has been a cornerstone for sharing and leveraging CI/CD pipelines of community, in-house and vendor software. Common tools such as repositories and artifactories have accelerated DevOps across teams. The second one is the awareness about reducing CapEx and OpEx using IaC and software-defined infrastructure technologies. As different recipes are created and shared across platforms for instantiating vClusters and configuring middleware such as WLCG, the deployment teams are identifying opportunities to maximise transferable tools and skills and minimise single points of failures and resource contentions, both human and technical. This has led to an increased simplification into CSCS technical portfolio without compromising the scalability of its service offerings. As an example, there were several storage solutions for POSIX-based parallel file system services for HPC and HTC clusters, fulfilling availability, performance and accessibility requirements. The multi-tenant vCluster model requires a multi-tenant file system to maintain isolation. The WLCG team introduced Ceph for dCache as a solution so that it can be extended, in a programmable manner, to other non-HPC use cases such as users home and application folders. This is an example of a configuration that was introduced for an HTC platform but can also be leveraged and reused for the HPC platform by adapting configuration changes.

Furthermore, vClusters enabled collaborative developments with the WLCG middleware development teams and facilitated integration of platform or VO representatives. One of the requirements is enabling VO representatives to track health and performance metrics in real time and on-demand. The cloud-native technology approach enables us to implement fine-grained access controls for different roles, which were not feasible on classic supercomputing platforms where there are two roles, an administrator and cluster users. Exceptions are typically created manually for middleware specific roles and services. This approach was not scalable and sustainable for diverse workflows, and was prone to security vulnerabilities. On Alps, WLCG tenants, monitoring and regression testing operations required by VO roles will be tracked using two mechanisms:

1. Setting a dedicated service or view to collect and show health and performance metrics and statistics. The specific contents will be determined over time with a feedback loop with the VOs.
2. Building a regression suite that can be used to verify functionality anytime. Operators run it before and after change management operations.

Table 2 lists the vCluster approach for two platforms on Alps dealing with diverse workflow scenarios, a classic HPC platform for running batch simulations and a data-driven Grid platform with HTC requirements. We identified common and distinct features because one platform (HPC) can be configured as per site needs, therefore, has a greater degree of freedom for selecting CI/CD pipelines and tools, while the WLCG software stack distribution is centrally controlled. The common aspects are build, test, stage and production tenants for individual platforms. For the complex grid middleware, we focus on the coverage of services, portability and extensibility, and operational efficiency using the vClusters approach. Using standard technologies like Ceph and Kubernetes for IaC, we have developed highly portable and extensible solutions, beyond Alps. This in turn allows developers, engineers and administrators to use common CI/CD pipelines and tools reducing overhead for distributed development teams and site-specific optimisations. The unique contribution of vClusters is development of multi-tenant slurm clusters that are used for the CE services (HTC payload). On the one hand, using an IaC approach, we can create multi-tenant slurm clusters that can have custom recipes without a single, shared slurm configuration constraint of a supercomputing HPC platform. On the other hand, we gain elasticity so that resource capacity can be modified dynamically, without change management overheads

Table 2. Key constraints for realising users and platform-driven CI/CD pipelines on contemporary supercomputing ecosystems. Typical HPC configurations are listed as examples highlighting the reasons why CI/CD pipelines and tooling that are commonplace in the mainstream software development teams cannot be exploited in a supercomputing system today. These configurations are compared with the requirements of mainstream CI/CD pipelines that are often a prerequisite for DevOps, enabling close-loop development, testing, delivery and deployment of software artefacts. Standard approaches are listed for addressing the requirements thereby enabling DevOps across the supercomputing ecosystems, specifically with proprietary hardware and software stacks such as the Cray-HPE Shasta supercomputer.

Constraint	Standard approaches	HPC platform	WLCG platform
1	Reusable and customisable programming environment and runtime configurations representing the relevant state of a production cluster that can be instantiated independently as multi-tenant, software-defined infrastructure (machines or clusters)	Programming environment is a configurable option within the HPC vCluster creation procedure. On-demand test and development vClusters can be instantiated by reusing and adapting existing, working recipes autonomously	not directly applicable because user software stack is centrally managed. DEV, STAGE and PROD vClusters support CI/CD pipelines to the supercomputing infrastructure
2	Reusable and customisable configuration images per cluster tenant that can be instantiated independently by platform developers as isolated tenants, for their CI/CD pipelines	CSCS HPC platform configuration is stored and updated using immutable IaC approach. Instances of vClusters can be generated as separate tenants on Alps for independent CI/CD pipelines	The worker node images are generated using WLCG containerised software stack as part of the image build process. DEV, STAGE and PROD zones
3	Decoupled infrastructure and platform provisioning using DevOps approach. CI/CD pipelines of different artefacts are independently instantiated and validated by reusing configuration recipes	Separate resources plane and infrastructure, platform and user stack management layers. The HPC vCluster stages specify the interfaces with the resource layers, for instance, requesting specific number and type of compute nodes and login nodes	WLCG has historically supported separation of site-specific and platform abstractions for infrastructure. For full coverage of the middleware, IaC on vCluster for Alps and Ceph (file system) have been used

and delays. These vClusters are resized for production, development and at-scale testing, without having to dedicate resources for specific operations. This in turn can minimise overall investment and operational overheads for different services because the infrastructure resources are part of a much bigger pool of resources, available to multiple tenants. On a regular basis, diverse teams can provision, deploy, test and destroy vClusters for their CI/CD pipelines on Alps resulting in autonomy of the teams and enabling DevOps in a supercomputing ecosystem.

Related work

Different public cloud providers have solutions for provisioning HPC clusters for batch scheduling systems like slurm that can be instantiated using software recipes on their platforms. AWS has (AWS ParallelCluster). Azure provides AzureBatch and (Azure CycleCloud). GCP has cloud HPC toolkit for configuring and deploying clusters (GCP Slurm Cluster). These solutions are similar to the vCluster approach such that infrastructure is defined in software, and therefore, can be managed as IaC. The integration targets are different where vClusters is a relatively narrowly scoped, open-source solution co-designed with the use cases for scientific platforms that represent diverse HPC and data-driven workflow characteristics. The public cloud solutions are integrated with respective ecosystems of individual providers, aligned with their respective business models. Moreover, due to the proprietary nature of these solutions, portability and extensibility to other resource providers and HPC data centres have not been addressed.

To address the proprietary abstraction, some research groups have developed solutions like (Cluster-in-the-Cloud) using standard abstractions such as Terraform (TerraForm), which is defined as an open-source IaC. Cluster in the cloud essentially leverages Terraform for deploying small-scale HPC clusters on public clouds. At the time of writing this article, Terraform did not have provider plugins for Cray-HPE Shasta systems. In future, vCluster work can potentially be leveraged to develop such plugins in collaboration with the supercomputer vendors. However, the current Shasta system management software stack and APIs do not have necessary abstractions to create such plugins. This could be an opportunity for not only Cray-HPE Shasta stack but vendors of other supercomputing technologies.

The usage and utilisation models for supercomputing, and public and private cloud services are considerably different and are not addressed within the scope of this article. However, in relation to the work presented in this article, multiple, isolated tenants are fully supported (compute, networking and storage) and resource can programmatically migrate across vClusters in case of underlying hardware failure. In fact, system and software stacks are decoupled and are patched and upgraded independently and continuously, using a so called cloud DevOps and Site Reliability Engineering (SRE) processes (Beyer et al., 2018).

Different open-source, close-to-metal, HPC solutions have been supported by public cloud providers, in form of batch HPC clusters for running applications or, in some cases, for bursting workload of an on-premise cluster (Tefler 2017). These are nevertheless not a replacement for a shared supercomputing ecosystem where a single job can potentially scale to the entire low-latency, high-performance networking domain, which can be defined as an RDMA (Remote Direct Memory Access) network scaling in contrast with typical L2 and L3 cloud network scaling (Alam et al., 2020). All compute resources of Alps supercomputer are connected via a low-latency, high-bandwidth proprietary interconnect technology called SlingShot (Khorassani et al., 2022). With the vCluster solution, we obtain native performance and scaling characteristics for large-scale MPI jobs, across the whole infrastructure and within individual tenants.

There are other approaches that have been investigated for finding optimal convergence of HPC and cloud technologies from a resource management and workload orchestration perspective. Flux introduces a hierarchical resource management concept that goes beyond a single cluster, like a meta-scheduler, and extends resource definition beyond a compute node (Ahn et al., 2014). This approach gives more control to the resource providers and users for requesting and managing resources but does not address the composibility issues for managing CI/CD of software artefacts. Elasticity is limited to the subnet of individual HPC clusters. Another approach that has been explored is called KubeFlux where HPC MPI jobs are run inside a Kubernetes cluster (Misale et al., 2021). This is an interesting approach as it keeps the cloud-native model to the infrastructure management layer. However, as reported in the article, Kubernetes is not natively designed for launching a large number of (MPI) tasks in a synchronous manner resulting in relatively significant job launch overheads. This is a promising approach but requires additional research to scale it to supercomputing platforms. Other hierarchical approaches included Apache Mesos that is used for resource management and scheduling of workloads across distributed data centre and cloud environments with dynamic resource sharing (Hindman et al., 2011). There have been efforts to extend HPC plugins for Mesos but there are no recent updates on integration and results on contemporary supercomputing platforms.

Grid middleware communities have been gradually adopting cloud-native technologies for the development and deployment across WLCG data centres. For instance, (Roy et al., 2020) demonstrated the management burden is reduced and uptime has increased by adopting cloud-native technologies such as containerisation and container orchestration systems. This work has similarities to our approach for adopting standard CI/CD tools. However, the scope of the work did not include unique requirements of a supercomputing ecosystem, which has been addressed by our approach. Presently, WLCG is providing singularity containers (Singularity), which is a container solution for HPC batch systems with shared file systems, to provide pilot/payload isolation and to run the payloads against the operating system of their choice, rather than that provided by the host site. There is a specific work group in WLCG targeting containerisation efforts. We can deploy Singularity as well as standard docker containers on our HPC ecosystem using a container runtime called (Sarus). The procedure for creating vClusters on Alps includes build of images for compute nodes and the application stacks allowing targeted container runtime to be introduced as part of the configuration recipes.

With the planned upgrades of the LHC and expected growth in the amount and complexity of data collected by its experiments, CERN's computing infrastructures will be facing a large and challenging demand of computing resources (CERN Tape Archive) (Fartoukh et al., 2021). Within this scope, the adoption of OpenStack cloud computing at CERN has opened the door to the evaluation of commercial cloud services, which could supply additional Infrastructure-as-a-Service (IaaS) resources to extend the current CERN computing resources for physics data processing (Cordeiro et al., 2017). While OpenStack is highly customisable for cloud-native workloads, it has not been adopted in HPC, especially for large, multi-Petascale systems running MPI applications at scale. One of the main reasons being overheads of virtualization that prevent large-scale MPI jobs requiring order of micro-second latencies and order of terabytes per second global bandwidths for communication. The multi-tenant vClusters on Alps rely on the Shasta system management stack that interfaces with the SlingShot fabric manager natively.

At the Oak Ridge National Laboratory (ORNL) leadership computing facility, a project was completed a few years ago titled: Big PanDA Workflow Management on Titan for High Energy and Nuclear Physics and for Future Extreme Scale Scientific Applications (De et al., 2021). The project aimed at exploring WLCG workflow models for future exascale computing by increasing the technical interoperability between HPC and that used for data-analytic, HTC usage patterns. The demonstrator project developed practical experiences and solutions to support diverse use cases and requirements, sharing workloads across HTC and

HPC resources, and investigating the operational implications of running traditional HTC workloads on HPC resources. The extensibility and portability of the solution was specific to the Titan supercomputing ecosystem for the targeted workflows of a given VO. Additional effort would be required to reproduce the solution for different workflows and on different supercomputing ecosystems. vCluster-based implementation of the WLCG platform is portable and extensible to other IaC compliant resource providers.

Conclusions and future directions

vCluster (versatile software-defined cluster), which is based on Infrastructure-as-code (IaC) technology, addresses the key constraints for enabling CI/CD pipelines of software artefacts on supercomputing ecosystems. The vClusters approach is a unique fusion of HPC and cloud technologies resulting in a software-defined, multi-tenant cluster on a supercomputing ecosystem, that, together with software-defined storage, enable DevOps for complex, data-driven workflows like grid middleware, alongside a classic HPC cluster. Specifically, vClusters address key constraints that prevent configuration of CI/CD pipelines of software artefacts on supercomputing ecosystems, which have bespoke software development, deployment, integration and operational stacks, with site-specific change management policies. The design principles maximise reuse of software artefacts and configuration recipes. By far, the most valuable byproduct is an expansion of knowledge and expertise of HPC and cloud technologies among the software development and infrastructure operation teams. The future development tasks include investments into automation, particularly configuration and operations of Kubernetes alongside multi-tenant slurm HPC clusters and support for standard interfaces like Terraform provider plugins for supercomputing platforms in order to leverage vCluster artefacts broadly, beyond the Alps supercomputing ecosystem. Furthermore, vClusters have opened up possibilities for innovating CI/CD pipelines of the user environments while addressing the security constraints of shared supercomputing ecosystems.

Acknowledgements

We would like to thank members of CSCS, the Swiss Institute of Particle Physics called CHiPP (all experiments and Virtual Organisation or VO representatives), and the Cray-HPE engineering and operational teams, and the CSCS user communities for their contributions and support for the development, deployment and evaluation of the underlying technological solutions that have been co-designed for as target use cases for vClusters.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iDs

Sadaf R Alam https://orcid.org/0000-0002-2534-5078
Maxime Martinasso https://orcid.org/0000-0003-1849-1621

References

ARC middleware. Available at: https://github.com/nordugrid/arc (accessed 13 March 2023).

CSCS Alps. CSCS introduction to Alps. Available at: https://www.cscs.ch/computers/alps (accessed 13 March 2023).

Artifactory. Artifactory - universal artifact repository manager. Available at: https://jfrog.com (accessed 13 March 2023).

AWS ParallelCluster. AWS ParallelCluster. Available at: https://aws.amazon.com/hpc/parallelcluster/(accessed 13 March 2023).

Azure CycleCloud. Azure CycleCloud. Available at: https://learn.microsoft.com/en-us/azure/cyclecloud (accessed 13 March 2023).

Cluster-in-the-Cloud. Create a cluster in the cloud. Available at: https://cluster-in-the-cloud.readthedocs.io (accessed 13 March 2023).

Ceph. Ceph documentation. Available at: https://docs.ceph.com/en/quincy/(accessed 13 March 2023).

CERN Tape Archive. CERN Tape Archive. Available at: https://cta.web.cern.ch/cta/(accessed 13 March 2023).

Cray-HPE Shasta EX. Cray exascale supercomputer. Available at: https://www.hpe.com/us/en/compute/hpc/supercomputing/cray-exascale-supercomputer.html (accessed 13 March 2023).

Cray System Management Cray HPE Shasta Software Stack. Available at: https://cug.org/proceedings/cug2019proceedings/includes/files/inv113s1-file1.pdf (accessed 13 March 2023).

Dcache. dCache virtual file system. Available at: https://github.com/dCache/dcache (accessed 13 March 2023).

Docker. Accelerated, containerized application development. Available at: https://www.docker.com (accessed 13 March 2023).

Frontier Supercomputer. Frontier - OLCF - Oak Ridge National Laboratory. Available at: https://www.olcf.ornl.gov/frontier (accessed 13 March 2023).

GitHub. Available at: https://github.com (accessed 13 March 2023).

GCP Slurm Cluster. Deploying a slurm cluster on compute engine. Available at: https://cloud.google.com/architecture/deploying-slurm-cluster-compute-engine (accessed 13 March 2023).

CSCS Eiger Cluster CSCS running on alps. Available at: https://user.cscs.ch/access/running/alps (accessed 13 March 2023).

LUMI Supercomputer. About LUMI - LUMI supercomputer. Available at: https://www.lumi-supercomputer.eu/about-lumi (accessed 13 March 2023).

Open Container Initiative. Open Container Initiative. Available at: https://opencontainers.org (accessed 13 March 2023).

OSU MPI benchmarks. OSU MPI micro-benchmarks. Available at: https://mvapich.cse.ohio-state.edu/benchmarks/ (accessed 13 March 2023).

IOR Benchmarks. IOR benchmark. Available at: https://wiki.lustre.org/IOR (accessed 13 March 2023).

HPE IaC Infrastructure-as-Code approach. Available at: https://www.hpe.com/sa/en/what-is/infrastructure-as-code.html (accessed 13 March 2023).

IaC Cattle Pet Analogy Cattle or Pet – what IaC means and why you shouldn't use admin-UI's. Available at: https://www.thesoftwareblog.com/2019/cattle-versus-pet-what-iac-means/ (accessed 13 March 2023).

Sarus The container engine for combining container portability with native HPC performance. Available at: https://products.cscs.ch/sarus/ (accessed 13 March 2023).

Singularity Singularity container platform. Available at: https://sylabs.io/singularity (accessed 13 March 2023).

ORNl Slate. ORNL slate service. Available at: https://docs.olcf.ornl.gov/services and applications/slate/index.html (accessed 13 March 2023).

TerraForm. Open source IaC: Terraform by HashiCorp. Available at: https://www.terraform.io (accessed 13 March 2023).

Top500. Top 500 list of fastest supercomputers. Available at: https://www.top500.org (accessed 13 March 2023).

WLCG. Worldwide LHC computing grid. Available at: https://wlcg.web.cern.ch/(accessed 13 March 2023).

AhnAhn DH, GarlickGarlick J, GrondonaGrondona M, et al. (2014). Flux: a next-generation resource management framework for large HPC centers. In 43rd international conference on parallel processing workshops, Minneapolis, MN, USA, 09-12 September 2014. DOI: 10.1109/ICPPW.2014.15.

AlamAlam SR, KleinKlein M, BiancoBianco M, et al. (2020). Software defined infrastructure for operational numerical weather prediction, communications in computer and information science. In Driving scientific and engineering discoveries through the convergence of HPC, big data and AI - 17th smoky mountains computational sciences and engineering conference, Oak Ridge, TN, USA, 2020. DOI: 10.1007/978-3-030-63393-6_20.

Allen BS, Ezell M, Peltz P, et al. (2020) Modernizing the HPC system software stack. Available at: https://arxiv.org/pdf/2007.10290.pdf (accessed 13 March 2023).

Beyer B, Murphy NR, Fong-Jones L, et al. (2018) How SRE relates to DevOps. Available at: https://www.safaribooksonline.com/library/view/how-sre-relates/9781492030645/(accessed 13 March 2023).

BoccaliBoccali T, CameronCameron D, CardoCardo N, et al. (2021). Dynamic distribution of high-rate data processing from CERN to remote HPC data centers. *Computing and Software for Big Science* 5: 7. DOI: 10.1007/s41781-020-00052-w.

Cordeiro C, Field L, Bear BG, et al. (2017) CERN computing in commercial clouds. *Journal of Physics: Conference Series* 898. DOI: 10.1088/1742-6596/898/8/082030.

De K, et al. (2021) BigPanDA workflow management on titan for energy and nuclear physics and for future extreme scale scientific applications. Available at: https://www.osti.gov/servlets/purl/1765084.

Fartoukh S, Kostoglou S, Solfaroli M, et al. (2021) LHC configuration and operational scenario for run 3. Available at: http://cds.cern.ch/record/2790409/files/CERN-ACC-2021-0007.pdf.

Gila M, et al. (2023) The WLCG journey at CSCS: from piz daint to alps. *To appear in the Cray User Group Proceedings*. https://cug.org/digital-library/

Heap M (2016) *Ansible: from beginner to pro*. 1st edition. New York, UK: Apress publisher. ISBN 1484216601.

Hightower K, Burns B and Beda J (2017) *Kubernetes: up and running dive into the future of infrastructure*. 1st ed. Sebastopol, CA: O'Reilly Media, Inc.

Hindman B, et al. (2011) Mesos: a platform for fine-grained resource sharing in the data center. *NSDI* 11: 22–22.

Jette MA, Yoo AB and Grondona M (2003) SLURM: Simple linux utility for resource management. *Lecture notes in computer science: proceedings of job scheduling strategies for parallel processing (JSSPP)*. Berlin, Germany: Springer-Verlag.

Khorassani KS, Chen CC, Ramesh B, et al. (2022) *High performance MPI over the slingshot interconnect: early experiences practice and experience in advanced research computing (PEARC '22)*. New York, NY, USA: Association for Computing Machinery, pp. 1–7. DOI: 10.1145/3491418.3530773.

Lundin C and Fisher S (2019) Significant advances in Cray system architecture for diagnostics, availability, resiliency and health. *Cray User Group Meeting*. Available at: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://cug.org/proceedings/cug2019_proceedings/includes/files/pap143s2-file1.pdf

Misale C, Daniel M, Maurizio D, et al. (2021). Towards standard kubernetes scheduling interfaces for converged computing. *Driving scientific and engineering discoveries through the integration of experiment, big data, and modeling and simulation*, Berlin, Germany: Springer.

Morris K (2016) *Infrastructure as code: managing servers in the cloud*. 1st ed. Sebastopol, CA: O'Reilly Media, Inc.

Roy G, Simili E, Stewart G, et al. (2020) Using Continuous Deployment techniques to manage software change at a WLCG Tier-2. *Journal of Physics: Conference Series* 1525. DOI: 10.1088/1742-6596/1525/1/012066.

Scheerman M, Zarrabi N, Kruiten M, et al. (2021). Secure platform for processing sensitive data on shared HPC systems. arXiv.2103.14679

Telfer S (2017) *The crossroads of cloud and HPC: openstack for scientific research: exploring openstack cloud computing for scientific workloads*. 1st ed1st edition. Scotts Valley, CA: CreateSpace Independent Publishing Platform.

Zhou N, Georgiou Y, Pospieszny M, et al. (2021) Container orchestration on HPC systems through Kubernetes. *Journal of Cloud Computing* 10.

Author biographies

Sadaf R Alam is director of strategy and academia in the Advanced Computing Research Centre at the University of Bristol. Previously, Dr Alam was a computer scientist at the Oak Ridge National Laboratory (ORNL) and ORNL leadership computing facility, and CTO the Swiss National Supercomputing Centre responsible for ensuring the end-to-end integrity of HPC and storage solutions, and leading strategic co-design initiatives. She is a member of the Association for Computing Machinery, ACM-Women, ACM's Special Interest Group on HPC and Women in HPC (founding member of the Swiss Chapter).

Miguel Gila is senior HPC System Engineer at the Swiss National Supercomputing Centre with broad experience operating and co-architecting HPC and HTC systems. With a strong interest in cloud technologies, he currently focuses in the development of the flagship Alps infrastructure at CSCS.

Mark Klein is head of the Systems Engineering unit and Systems Architect at the Swiss National Supercomputing Centre operated by ETH Zurich in Lugano. Previously, he has worked on the Blue Waters project at the National Center for Supercomputing Applications and in the Scalable Computing Lab at the US DOE's Ames Laboratory. He led technical implementation of several co-design projects including the flagship supercomputing and operational weather forecasting systems at CSCS.

Maxime Martinasso is the Head of Engineering at the Swiss National Supercomputing Centre where he oversees the identification, planning, and execution of engineering developments in HPC technology and infrastructure, with a current focus on the convergence of Cloud and HPC, AI and data management. Before his current role, Dr. Martinasso led several European projects and HPC technical teams in different fields and industry. He holds a PhD in the field of HPC.

Thomas C Schulthess is director of the Swiss National Supercomputing Centre (CSCS) and a professor for computational physics at ETH Zürich. He received his PhD in 1994 from ETH Zürich and spent many years at Oak Ridge National Laboratory, where today he holds a distinguished visiting scientist appointment. While his primary research is on computational methods for materials science, he recently took interest in the development of energy-efficient computing systems for climate modelling and meteorology.