 doi: 10.15496/publikation-29062

Game of Templates

Deploying and (re-)using Virtualized Research Environments in High-Performance and High-Throughput Computing

Jonathan Bauer* Dirk von Suchodoletz* Jeannette Vollmer*
Helena Rasche†

*eScience Department, Computer Center, University of Freiburg, Freiburg, Germany
†Department of Bioinformatics, University of Freiburg, Germany

The Virtual Open Science Collaboration Environment project worked on different use cases to evaluate the necessary steps for virtualization or containerization especially when considering the external dependencies of digital workflows. Virtualized Research Environments (VRE) can both help to broaden the user base of an HPC cluster like NEMO and offer new forms of packaging scientific workflows as well as managing software stacks. The eResearch initiative on VREs sponsored by the state of Baden-Württemberg provided the necessary framework for both the researchers of various disciplines as well as the providers of (large-scale) compute infrastructures to define future operational models of HPC clusters and scientific clouds. In daily operations, VREs running on virtualization or containerization technologies such as OpenStack or Singularity help to disentangle the responsibilities regarding the software stacks needed to fulfill a certain task. Nevertheless, the reproduction of VREs as well as the provisioning of research data to be computed and stored afterward creates a couple of challenges which need to be solved beyond the traditional scientific computing models.

1 Motivation

The exponential growth of computational power in the past decades has greatly contributed to scientific advances in all fields. One of the key success strategies in

CC BY 4.0 doi: 10.15496/publikation-29057

science is to recognize recurring patterns and exploit them via templates. First, find out which part of a problem is static or invariant – this becomes the template. Then iterate over the variable part of the problem to search for the solution.

The development of hardware virtualization for the x86 platform in the last two decades and the cloud revolution also triggered a paradigm shift for university computer centers. The way IT resources are provided and which services should accompany them is changing. The ubiquitous use of digitalized workflows and the Fourth Paradigm in science demand an ever-increasing amount and variety of IT-based research infrastructures. To avoid handing over sizeable proportions of infrastructure-providing activities to the commercial domain – for reasons ranging from privacy and security to expertise considerations – computer centers have to find new ways to offer a significant range of infrastructures in an efficient way. It should provide comparable offerings regarding features and pricing[1] as well as to avoid overextending existing personnel resources when scaling up. Demands for hardware often come up on short notice and for project periods well below the cost-amortization period of five to six years that is typical for digital equipment. Having decentralized and often duplicated personnel to select, procure and operate all the various research infrastructure components is expensive.

Further challenges of university computer centers and faculty IT units are rooted in the very diversity of scientific communities and their broad set of demands with respect to software, tools or scientific workflows. This creates varied and often contradicting demands regarding software environments. Facilitating virtualization can help to separate the different requirements. As many resources in research infrastructures are underutilized for certain time periods, tapping into cloud strategies can help to significantly save on investment and hardware resources. A welcomed by-product would be savings on rackspace and energy.

2 Project objectives and related work

The Virtual Open Science Collaboration Environment (ViCE) project – sponsored by the state of Baden-Württemberg under the umbrella of the eScience initiative – brought together researchers from various science domains and infrastructure providers (computer centers from different universities). Its goal was to facilitate the

[1] The term »pricing« is used in a wider sense here, as it is necessary to consider different models in basic free services, cost recovery or extension of infrastructure by bringing in project money.

exchange of ideas, concentrating on the separation of the responsibilities of infrastructure providers from core scientific tasks and vice versa. Virtualized Research Environments (VRE) are a core concept to achieve a separation of tasks while preserving flexibility on the sides of both the users and providers.

Research projects should enjoy a quick start without tedious workflows to procure and set up the necessary IT infrastructure. Especially compute resources need to scale up and down following the demands of the individual project progress. At the same time, students and research assistants need to be integrated efficiently into research workflows. Virtual Machines (VM) can help by allowing prepared software environments to be copied, avoiding setting up the complete hardware, operating system, and application stack including configuration. Additionally, individual researchers and workgroups should gain more flexibility to set up their own derived versions of research environments and workflows.

The ViCE project aimed to loosen the originally tight connection matrix between research, administration, hardware, and software. A less-static environment invalidates a couple of traditional assumptions: NFS IP-based authentication becomes less an option with the dependence on local authentication frameworks. If VMs are really moved around to be used on different platforms and on different sites, deployment workflows have to be adapted. This will provide new means for experimenting and exchanging ideas and (complete) digital workflow environments.

The term Virtual(ized) Research Environment appears in the context of eResearch and eResearch infrastructures. VREs were introduced to foster cooperation in distributed projects and the exchange of complete software stacks. The shared resources not only mean shared data but additionally shared scientific workflows. Different VREs may focus on different aspects like versioning or large scale distribution and may come in different forms of representation. The author of (Allan, 2009) gives a wide-ranging definition of VREs in various forms by describing them in terms of intended capabilities.[2] Many projects saw VREs focused on web-based access to resources, though. »myExperiment« (De Roure et al., 2009) is a large public repository of scientific workflows created e. g. with Taverna (Oinn et al., 2006) or Galaxy (Afgan et al., 2018). It offers a collaborative environment where researchers can share publish and cite published scientific workflows. Workflows can be packed with digital objects to be swapped, sorted and searched. Taverna and Galaxy are pro-

[2] See p. 11f. VRE description includes different modes of operation from desktop to servers, talks of means of accessibility and usability, workflows and the focus on Open Source.

jects that enable users to graphically compose bioinformatics (and other) workflows exploiting several web services and tools spread all over the world. It allows the design, development and execution of scientific workflows over an existing compute and storage (science grid) infrastructure. The workflows are written in SCUFL and have a particular XML schema. The FreeFluo workflow engine, which is coupled with Taverna, manages the execution of the workflows. While Taverna and Galaxy focus on high-level workflows and existing execution environments, VREs in the context of ViCE focus on the software environment powering the workflows (Meier et al., 2017). The approach of using a VM consisting of the full software stack is suitable to allow a wide range of VRE variants of different disciplines, as workflows can be represented by more than just web-enabled tools.

VREs are packaged software tools for data analysis and computing which together with the research data processed through them represent the complete scientific workflows. To make them findable like data sets, a repository and registry are needed. These could be used as well for versioning or for exchange of VREs among researchers.

To make VREs more common and to attract a wider user base, different hosting platforms should be enabled to accept VREs. For various purposes in research and teaching, those platforms range from the desktop environment used in preparation of workflows and for interactive teaching, to HPC and cloud infrastructures. The relevant platforms used throughout the project were primarily the bwForCluster NEMO, the bwCloud and bwLehrpool.[3] Thus, various container and virtualization technologies such as Singularity, Docker or OpenStack were enabled and evaluated on the relevant science infrastructure platforms.[4]

Further objectives of the project were to evaluate frameworks for the exchange of VREs between the computer centers of the state universities. It should allow joint teams to develop, test and deploy VREs of various forms and provide necessary provisioning workflows. Further, measures to provide secure computing environments for sensitive data were to be explored. Additionally, the findings of the project were to be communicated by various workshops and trainings.

[3] NEMO (`http://www.hpc.uni-freiburg.de/nemo`), bwCloud (`https://www.bw-cloud.org`) and bwLehrpool (`https://www.bwlehrpool.de`) are federated state-wide infrastructure projects co-financed by the Ministry of Science, Research and the Arts, Baden-Württemberg in different configuration of partners and scope (visited on 05. 01. 2019).

[4] See `https://singularity.lbl.gov`, `https://www.docker.com` and `https://www.openstack.org/` (visited on 11. 01. 2019).

3 Steps to create a VRE

Building on the purpose and findings from previous projects, VREs can support highly differentiated goals, from virtualizing environments which were meant to be installed on compute clusters to packaging certain tools to be accessed via web services. They are usually tailored to the scientific application and can take the form of templates which are adapted by single researchers or cloned for massive parallelization. VREs can help to manage complexity by splitting workflow steps into distinct machines with clearly defined purposes which can later be chained or re-used in different workflows. The complexity of steps to complete will vary depending on the purpose of the VRE:

1. Audience: Depending on the intended purpose, VREs are meant to be created and run per scientist, per scientific workgroup, or even per scientific field.[5] Options of authentication depend on the actual users to log-on to the machine, if any. The computation of sensitive data e. g. person-related information might be restricted on certain infrastructures as these might not comply with all requirements.
2. Define the amount of resources: The requested resources for scientific workflows might differ significantly and may require grid systems, resource brokers, portals, knowledge systems or (large) data collections to be included.
3. Technical environment: The origin technical platforms may include desktop, cloud or HPC resources defined by the intended use e. g. for tool and workflow development, teaching and learning or non-interactive massive computation. Access to special hardware resources might be required.
4. External dependencies of filesystems, identity management systems or, if required, of license servers and the like needs to be resolved.
5. Planning of setup and maintenance: Optimally, instances can be created by automatic procedures like Packer, Ansible, Puppet and similar or can be cloned from templates.[6] Long-running VREs might require updates and older versions may need to be stored for reproducibility.

[5] In the scope of the ViCE project different variants were evaluated.

[6] See `https://packer.io/`, `https://www.ansible.com/`, `https://puppet.com` (visited on 12. 01. 2019).

4 Experiments and findings

During the project several use cases were evaluated ranging from bioinformatics VREs in the form of Galaxy services, two different types of particle physics workflows (Meier, 2017; Bührer et al., 2018), and a range of unique VREs created for English language studies, economics, microsystems technology or neuroscience. The focus for the particle physics VREs rested on the complete reproduction of the well-defined software stack needed for the analyses run in the CME and ATLAS experiments based on the CERN VM, which is itself a Scientific Linux version 6. The VRE ensured the complete control over all software components including kernel and base libraries required for large-scale distributed experiments. Thus, a solution including full virtualization was required and containerization was not an option because of Linux kernel dependencies. The Galaxy VREs are meant to run bioinformatic workflow tools and are less dependent on the basic software layer. To use the same deployment workflows as for the particle physics environments, full virtualization was used in the beginning of the project. ViCE helped to enable containerization on HPC and the PC pool environments and make special container VMs available in the cloud. This allowed smooth migrations from one environment into the other and the same VRE could be demonstrated and interactively used in teaching lessons as well as non-interactively for mass computing as shown in Figure 1.

The ATLAS and CMS VREs as well as the workflow VMs are created for potentially massive parallelization and not meant to be accessed directly. Special users are created which run the relevant tools and offer the necessary interfaces to interact with them. The VREs are meant to be created or cloned and thrown away after use. For accounting purposes, they are assigned to the user who started them either via HPC job control or through the cloud API. The VREs created for the language studies was meant to be used by students and lecturers throughout the state of Baden-Württemberg and utilize Shibboleth authentication backed by the bwIDM federation.[7]

[7] Shibboleth based identity federation, `https://www.bwidm.de` (visited on 14. 01. 2019).

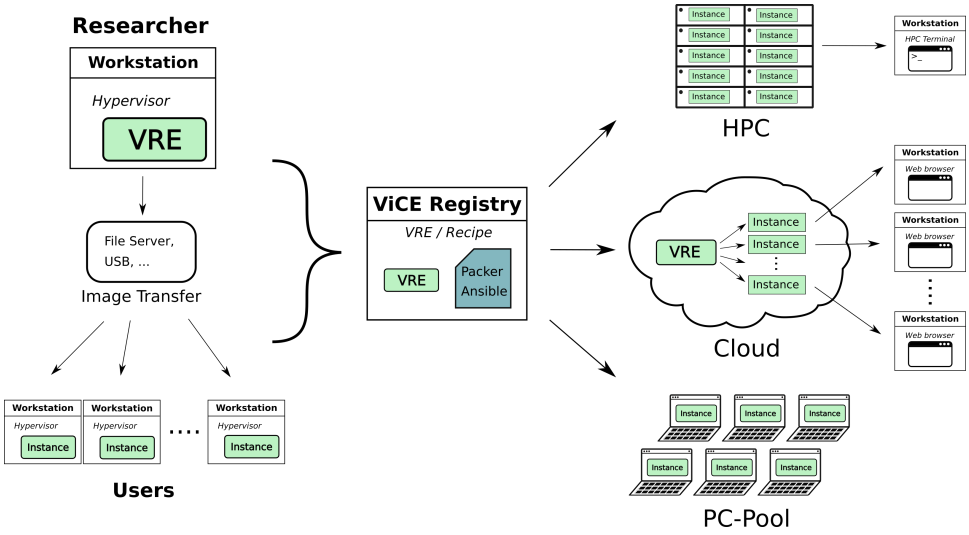

Figure 1: Reproducible VRE templates for flexible distribution.

4.1 Security considerations

The computation of sensitive data becomes even more a challenge in a VRE. The complete software stack including the hypervisor needs to be protected against compromise (Lombardi et al., 2011). While the topics of data ownership and quality of service are less of a concern in the compute environments considered during the project, confidentiality, integrity, data mobility and data protection remain significant challenges (Zissis et al., 2012; Shahzad, 2014). Encryption of data can provide a solution to secure storage when flexible access, scalability in key management and efficient user revocation are properly implemented (Li et al., 2013; Wang et al., 2012). But, necessary cryptographic operations lead to an additional complexity in cloud environments compared to traditional bare-metal environments. It is due to control of systems on which both the key management system and protected resources are located as well as difference in data owners and service providers (Chandramouli et al., 2014). Many of the challenges named require additional auditing and certificate infrastructure, rigorous processes by the infrastructure provider and (external) certification (Zissis et al., 2012).

To enforce data security and privacy different encryption techniques are deployed such as full disk encryption or fully homomorphic encryption. While the first encrypts the entire disk, the latter encrypts particular functions and is used to secure

data from exploitation during computation (Zhao et al., 2014). The first option was implemented as a baseline measure by encrypting the storage holding either the virtual machine images in the bwCloud or local scratch space in bwLehrpool. The implementation of concepts like homomorphic encryption is much more expensive to set up and requires additional computation (Tari et al., 2015). It was not further considered as it lay well beyond the primary scope of the ViCE project. Up to now, trusted computation and storage can not be guaranteed as the requirements can only partly be met in the existing infrastructure provided by NEMO or bwCloud at the Freiburg site.

4.2 Technical environment

After defining the intended purpose of a VRE, the technical specifications must be set: The software environment representing a certain scientific workflow may require a specific system environment such as kernel and system libraries with clearly defined versions. While full system virtualization using hypervisors reproduces complete machines, containerization tools such as Docker or Singularity use concepts like namespaces to separate environments. The overhead of e. g. CPU and IO virtualization and thus the potential loss in performance of the former case is heavier. In the latter case, the software of the VRE runs directly on the host kernel and is thus dependent on its version and capabilities. Full virtualization abstracts core hardware components and peripherals. Special purpose hardware like GPGPUs, Infiniband or Omni-Path infrastructures are not easily virtualized and not easily available from inside the VRE. They cannot be trivially shared among VREs running on a single host system, although there exist a couple of ways to dedicate such resources to single VM instances. Nevertheless, a fully virtualized VRE is less dependent on the existence of hardware components and thus easier to share and move across different host systems. Further challenges arise for tasks such as (remote) visualization of data as envisioned for a microsystem technology VRE.

To allow the sharing of GPU resources within the NEMO HPC cluster, a Docker or Singularity container was created which allows direct access to the necessary hardware and to the parallel file system at the same time. PCI passthrough is one of the options to allow VMs to access hardware in the host system, but exclusively. Nevertheless, it can help to share a well-equipped GPU node among completely different users and their software environments. Up to now, the experiments with

Docker and Nvidia GPU demonstrated a couple of kernel and driver challenges as software versions need to be tightly matched in the host and Docker environments, reintroducing dependencies meant to be overcome by virtualization.

4.3 External dependencies

Software and infrastructural dependencies become explicit if deployed in a VRE. One of the resources a research or teaching project might need is storage. Often source and destination shares, e. g. home directories, or software module collections are mounted from a central resource and secured by defining IP ranges to which an export is allowed. If used in a VRE, especially on top of different resources at different sites or if meant to be shared among different colleagues in a distributed group, this option is no longer suitable. A similar problem arises from latencies if the shared resource is not available from within the hosting site but a couple of network hops away. Moreover, higher latencies with jitter usually hurt the performance of traditional file systems.

SDS@hd (Baumann et al., 2017) is one service offering storage for projects in Baden-Württemberg and was tested to see if it was useful as a solution to the dependency problem. SDS@hd offers storage statewide to scientists at public higher education institutions.[8] The service specifies that the storage is intended for data in active use, not for long term storage or backups. This means that it could be useful in exactly the cases outlined above – cloned or parallel projects requiring access to shared data or a space to save their results.

Conveniently, SDS@hd also offers an existing test project that can quickly and easily be connected to in order to determine if the service will work for a specific project or in the given infrastructure. For a productive use of this service an entitlement must be granted: first an entitlement by the institute, then a request for a specific amount of storage with justification must be submitted; after receiving provisional approval, a contract must be signed and submitted before the allocation can be approved. This process has to be completed once for every storage project, but once it is done the project owner can easily invite other users to the partition.

Once approved, the storage project could be accessed by various methods – SSHFS access was easy and instantly available using a password of one's own choosing; NFSv4 access required human interaction (providing personal data and information

[8] Subsidised by the university for researchers in Heidelberg, at a fee for external users.

regarding the machine that would be used to make the connection) in order to generate a keytab for access; SMB is also a connection option, but was not tested in the course of the ViCE Project. Having heard complaints of slow data transfers with SSHFS as a potential negative outweighing the ease of connection, several tests were run to compare performance. While initial results confirmed the assumption that NFS would be faster, further tests were run using different ciphers, resulting in comparable results using both connection types. Tests showed that from bwCloud to the SDS@hd storage project, NFS was able to handle writes faster than SSHFS, and the inverse was true for reads. Thus, a good understanding of the usage patterns for each project and some preparation at setup time can pay off in the longer term for a project with intensive reads or writes.

4.4 Setup and maintenance of VREs

VREs are intended to exist over a long period of time, and to allow for reproducibly running software within the environments. Provisioning different types of VREs with their respective software stack while providing a generic deployment process to make the resulting images VM or container compliant requires a structured workflow. As such, the long term development and maintenance can become a significant concern. In order to combat these issues, VREs should be well-defined environments, from the base image to any changes applied to them. Any solution identified should provide a flexible base infrastructure provisioning, allowing for easy adaptation to any future scientific workflows. The challenge is to standardize as much of the process as possible, minimizing the efforts required to realize various software environments.

Leveraging infrastructure-as-code is a solution to this problem; by defining VREs as a base image and a set of provisioning steps, managed in a git repository and output as a bootable machine image, the process of developing and deploying reproducible, re-usable infrastructure is significantly simplified.

The basic VM installation is handled by Packer using an appropriate source image (e. g. minimal core distribution ISO or a cloud image). Complete use-case-specific software stack installation is then performed using plug-in software provisioning tools (e. g. Ansible or Puppet) creating the different image variants which will be used by downstream compute infrastructure as system images.

The building of these images for bare-metal and cloud consumption is done automatically via a Jenkins[9] continuous integration server. When changes are pushed to their respective git repositories, builds are automatically triggered. The resultant bootable images are either self-contained in the cloud use case or in addition to the kernel and generated using Dracut[10] initramfs for the bare-metal use case, are then deployed to iPXE boot servers or directly uploaded to OpenStack, ready for deployment.

Ansible and Packer were explored to allow for easily building and re-building of VREs as requirements evolved and bugs were discovered over time. Packer was used to boot an image, run provisioning steps within the VM, and save the output as a new image. The choice of Packer provided easy flexibility for deployment of varied software stacks on top of the base VMs; it permitted a choice in virtualisation method (QEMU, VMware, numerous cloud providers), and a choice in provisioning method (shell scripts, Puppet, Ansible). Many pre-existing workflows leveraged Ansible playbooks to configure bare-metal nodes. Packer allowed for re-use of these existing provisioning workflows to directly create bootable machine images identical to their bare-metal counterparts.

A side benefit of this reproducible build of machine images is that, due to infrastructure existing as code in a git repository, it becomes possible to track the history and to reproduce old versions of images, which can be rebuilt and redeployed as needed.

The final result of this system is a clean division of labor where network boot and cluster administrators can each focus on their own tasks, independently of each other. With the automated image building and deployment process, we achieve the ability to rapidly produce VREs targeting the needs of specific users and communities with only small changes in configuration.

The base image template is utilized by various Ansible playbooks for different image flavors. The burden of VRE development is reduced by the playbooks which can be collaboratively used for basic tasks such as installing Singularity, updating packages, and managing services. Any future workflows that require additional software or services can be accommodated by forking the base image playbooks and by

[9] See `https://www.jenkins.io/` (visited on 01. 02. 2019).

[10] See `https://dracut.wiki.kernel.org/index.php/Main_Page` (visited on 05. 02. 2019) extended by a customized network boot module to enable stateless operations. Refer to (Schmelzer et al., 2014) for HPC stateless deployment in general.

implementing the new requirements. Any historical workflows can be reproduced by deploying an old image from the registry.

4.5 VRE registry

The experience with different scientific communities during the ViCE project has shown that software stacks or tool chains required for particular tasks are often similar across different workgroups within the same research field. As such, collaboration and exchange of VREs needs to be promoted to avoid individual groups creating similar VREs. Instead, the focus should be on sharing, improving and reusing existing VREs. To this end, scientists should have a way to search through available environments to find suitable VREs for their workflows. Finally, VREs should become part of Research Data Repositories (Pampel et al., 2013).

Gathering various categories of metadata is key to cataloging any kind of data and make it findable by others. From a user perspective, attributes like the specific research field, the operating system and the available software stack help identify relevant VREs for a particular purpose. From a technical perspective, minimal virtual hardware requirements, disk image or container format, are required to properly deploy them in various execution environments. Finally, from an operational perspective, management metadata like the researcher and the affiliated research project are useful for accounting and VRE life-cycle purposes. Before publication of scientific findings, archiving the employed software environment in order to reproduce the results is just as important as archiving the data sets utilized. This is essential for the long-term perspective of reproducible and citable research (Rechert et al., 2017).

To fill in these gaps, the idea of a central collaborative platform for users to manage, exchange and publish VREs was envisioned. A proof-of-concept implementation, the ViCE Registry, was quickly developed. As a first step, interfaces to OpenStack cloud infrastructures and PC pool infrastructure[11] were integrated and allowed users to import and export VREs from one system into the other and to search through those available in the registry (Hauser et al., 2017). While the prototypical implementation was promising at first, it was soon clear that the completion of all its features as well as its maintenance over time would not be sustainable in

[11] Orchestrated by the network booting bwLehrpool project (Suchodoletz et al., 2014).

the long-term. As such, it was discontinued and alternative concepts to realise an exchange and collaboration platform for VREs needed to be evaluated.

Some concepts in research data management, like the FAIR[12] principle, identified requirements similar to those of the ViCE registry (Wilkinson et al., 2016). From its early conception, the ViCE registry focused on those four requirements. Classifying metadata into organisational, cataloging and scientific attributes is essential for indexing purposes. Further, interoperability between computational research infrastructures allow researchers to access and reuse VREs in a diverse manner. Finally, reliable storage of the curated data and its metadata achieves its long-term preservation and accessibility. This led to a new ViCE registry concept melding into existing data management concepts, only focusing on the particular challenge of migrating VREs between different execution platforms.

A first evaluation of the data management software iRODS was promising (iRODS Consortium, 2017). Beyond the core storage functions and the large support of storage systems that enables custom hierarchical storage management, its ruling engine seems particularly attractive. Automatic gathering of organizational metadata on data ingestion depending on its origin, enforcing a set of metadata before publication of the data in a collaborative exchange area (facilitating the indexing thereof) or automation of VRE imports to and exports from computational research infrastructures are but a few examples of applications of iRODS rules (Rajasekar et al., 2010).

As an early experiment, the local PC pool's internal image exchange mechanism was adapted to use iRODS as a storage backend. Descriptive and technical metadata were inferred from bwLehrpool's internal data structure. Using iRODS metadata query language, third-party applications can search through the available images by criteria. For example, an OpenStack application using the iRODS API to search for VREs by technical metadata and then import compatible VREs into the cloud image repositories. Going a step further, the cloud image repository could use iRODS as a backend directly – then using a PC pool VRE in the cloud infrastructure would only require creation of a cloud image using image stored in iRODS, removing the need to actively transfer the image from one storage backend to another. These experiments were promising and confirmed the validity of an iRODS-based approach to create a VRE exchange platform. However, the documentation of the

[12] **FAIR** stands for **F**indability, **A**ccessibility, **I**nteroperability and **R**eusability.

iRODS API was lackluster. As such, it impeded the development of the interface to bwLehrpool's infrastructure and it can be expected that it will hinder its integration in the workflows of scientific communities. There are alternatives though, for example the object storage protocol S3 which offers multiple convenient interfaces to access its resources, such as a REST API and even s3fs [13], a third-party POSIX-like filesystem based on FUSE. Future work should evaluate an S3-based approach to realize a VRE registry and the extent to which an S3 storage layer can be complemented with a higher level iRODS layer for metadata management and automation, combining the best of both worlds.

5 Conclusion

The increased complexity of scientific workflows, the rising demands of researchers on compute power, and the sheer number of servers to monitor and administer demand new operation models. The workflows to run VREs are rather complex and took a while to mature. The first versions of VRE were in production pretty much since the official start of the NEMO cluster in mid 2016. Since then, a couple of improvements were implemented, but envisioned features like mapping Moab commands to OpenStack API allowing the pausing, hibernation and resumption of the virtual machine for preemption or maintenance instead of killing a job are still waiting to be tackled.

Limitations still exist for generalizing the use of VREs. The use cases considered featured embarrassingly parallel High-Throughput-Computing (HTC) workloads. These do not require high-speed low-latency networks for interaction between cluster nodes at all. Concentrating on such VREs simplified the setup and operation of virtual machines as special direct hardware access could be ignored. As further use cases like remote visualization emerge which require access to special hardware from inside a VRE, direct hardware access will be reconsidered in future activities. Experiments on containerization with Singularity show encouraging results for extending the scope of VRE deployment. A couple of bwHPC clusters plan to include Singularity by default in the near future. Having VREs in place opens up future paths like cloud-bursting.

[13] See `https://github.com/s3fs-fuse/s3fs-fuse` (visited on 10. 02. 2019).

A further challenge arises from the need to access data from within VREs. Traditional parallelized high-performance storage often does not provide the necessary security concepts to be directly accessed from within clearly defined security perimeters of a HPC system. If users become root in their VREs then the traditional means of privilege separation will no longer work. Another challenging issue arises from the intended heightened mobility of VREs. While in static cluster configurations, IP-based security even with its limitations made sense; the steps necessary to mount remote network filesystems into a VRE meant to run in more than one location no longer do. Even modern implementations like NFSv4 using account-based security face limitations. If a researcher moves on to another workgroup the account may be disabled and the access to the share becomes impossible. In the highly volatile environment of research institutions the chances for long-term stable user account-based access methods are dim. VREs become truly independent of location when the ties to traditional network and parallel file systems can be overcome e. g. by object storage solutions and access management gets moved on access tokens as well as global identities for researchers.

Depending on the way virtualization or containerization is orchestrated, the scheduling setup for the compute clusters has to be aware of the more dynamic nature of resources. HPC schedulers have to be aware of virtualized resources which, as a VM or Singularity container, are partially opaque to them. Further on when pre- and post-processing tasks in modern workflows are considered, these often profit from interactive handling instead of batch-driven automatic processing. Here, VREs could help to use the same working environment for cloud (pre-, post-processing and visualization) and HPC systems (main computational task). Containerization and VRE open the way to achieve Certified Research Environments in the sense that states of actual workflows could get frozen and archived in a consistent state. It would be beneficial to provide a platform and registry for researchers to get an overview of existing VREs including their relevant metadata. An ongoing challenge is the handling of sensitive data on shared resources like HPC and cloud. The requirements of the implemented data protection ruling are to be honored. Nevertheless, further research into versioning and long-term access to previous versions is still needed.

On the operational side, VREs allow the simple and convenient redistribution of tasks between the researchers focusing on the application side and the computer

centers focusing on providing scalable research infrastructure. It extends the chosen path of the successful centralization and specialization of HPC resources. It helps to easily provision additional hardware resources brought in by a third party. For the next generation bwHPC cluster in Freiburg, the HPC team will reconsider the options to reduce the complexity of the VRE scheduling. For the upcoming cluster, a distinct cloud partition for HTC-focusing VREs is planned. Additionally, new models for scaling and configuring the computational resources are evaluated (Bauer et al., 2019).

Acknowledgement

The work presented in this publication was a part of the ViCE project sponsored by the Ministry of Science, Research and the Arts, Baden-Württemberg, Germany. The support is gratefully acknowledged.

Corresponding author

Jonathan Bauer: jonathan.bauer@rz.uni-freiburg.de
eScience Department, Computer Center, University of Freiburg
Hermann-Herder-Str. 10, 79104 Freiburg, Germany

ORCID

Jonathan Bauer https://orcid.org/0000-0002-5624-2055
Dirk von Suchodoletz https://orcid.org/0000-0002-4382-5104
Jeannette Vollmer https://orcid.org/0000-0001-5190-2942
Helena Rasche https://orcid.org/0000-0001-9760-8992

License CC BY SA 4.0 https://creativecommons.org/licenses/by-sa/4.0

References

Afgan, E. et al. (2018). »The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update«. In: *Nucleic Acids Research* 46.W1, W537–W544. DOI: 10.1093/nar/gky379.

Allan, R. N. (2009). *Virtual research environments: From portals to science gateways*. Elsevier.

Bauer, J. et al. (2019). »A Sorting Hat For Clusters. Dynamic Provisioning of Compute Nodes for Colocated Large Scale Computational Research Infrastructures«. In: *Proceedings of the 5th bwHPC Symposium. HPC Activities in Baden-Württemberg*. Freiburg, September 2018. 5th bwHPC Symposium. Ed. by M. Janczyk, D. von Suchodoletz and B. Wiebelt. TLP, Tübingen, pp. 217–229. DOI: 10.15496/publikation-29055.

Baumann, M., V. Heuveline, O. Mattes, S. Richling and S. Siebler (2017). »SDS@hd–Scientific Data Storage«. In: *Proceedings of the 4th bwHPC Symposium October 4th, 2017, Alte Aula Eberhard Karls Universität Tübingen*. Ed. by J. Krüger and T. Walter, pp. 32–36. DOI: 10.15496/publikation-25204.

Bührer, F. et al. (2018). »Dynamic Virtualized Deployment of Particle Physics Environments on a High Performance Computing Cluster«. In: *Computing and Software for Big Science*. arXiv: 1812.11044 [physics.comp-ph].

Chandramouli, R., M. Iorga and S. Chokhani (2014). »Cryptographic key management issues and challenges in cloud services«. In: *Secure Cloud Computing*. Springer, pp. 1–30.

De Roure, D., C. Goble and R. Stevens (2009). »The design and realisation of the myExperiment Virtual Research Environment for social sharing of workflows«. In: *Future Generation Computer Systems* 25.5, pp. 561–567. DOI: 10.1016/j.future.2008.06.010.

Hauser, C. B. and J. Domaschka (2017). »ViCE Registry: An Image Registry for Virtual Collaborative Environments«. In: *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 82–89. DOI: 10.1109/CloudCom.2017.11.

iRODS Consortium, ed. (2017). *iRODS User Group Meeting 2017 Proceedings – 9th Annual Conference Summary*. 83 pp. URL: https://irods.org/uploads/2017/irods_ugm2017_proceedings.pdf.

Li, M., S. Yu, Y. Zheng, K. Ren and W. Lou (2013). »Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption«. In: *IEEE transactions on parallel and distributed systems* 24.1, pp. 131–143.

Lombardi, F. and R. D. Pietro (2011). »Secure virtualization for cloud computing«. In: *Journal of Network and Computer Applications* 34.4. Advanced Topics in Cloud Computing, pp. 1113–1122. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2010.06.008.

Meier, K. (2017). »Infrastrukturkonzepte für virtualisierte wissenschaftliche Forschungsumgebungen«. PhD thesis. Albert-Ludwigs-Universität Freiburg im Breisgau.

Meier, K., B. Grüning, C. Blank, M. Janczyk and D. von Suchodoletz (2017). »Virtualisierte wissenschaftliche Forschungsumgebungen und die zukünftige Rolle der Rechen-

zentren«. In: *10. DFN-Forum Kommunikationstechnologien, 30.-31. Mai 2017, Berlin, Gesellschaft für Informatik eV (GI)*, pp. 145–154.

Oinn, T. et al. (2006). »Taverna: lessons in creating a workflow environment for the life sciences«. In: *Concurrency and Computation: Practice and Experience* 18.10, pp. 1067–1100.

Pampel, H. et al. (2013). »Making Research Data Repositories Visible: The re3data.org Registry.« In: *PLOS ONE* 8.11. DOI: 10.1371/journal.pone.0078080.

Rajasekar, A. et al. (2010). »iRODS primer: integrated rule-oriented data system«. In: *Synthesis Lectures on Information Concepts, Retrieval, and Services* 2.1, pp. 1–143.

Rechert, K. et al. (2017). »Preserving Containers«. In: *E-Science-Tage 2017: Forschungsdaten managen*. Ed. by J. Kratzke and V. Heuveline. Heidelberg: heiBOOKS, pp. 143–151. DOI: 10.11588/heibooks.285.377.

Schmelzer, S., D. von Suchodoletz, M. Janczyk and G. Schneider (2014). »Flexible Cluster Node Provisioning in a Distributed Environment«. German. In: *Hochleistungsrechnen in Baden-Württemberg. Ausgewählte Aktivitäten im bwGRiD 2012*. Ed. by J. C. Schulz and S. Hermann. KIT Scientific Publishing, Karlsruhe, pp. 203–219. ISBN: 978-3-7315-0196-1. DOI: 10.5445/KSP/1000039516. URN: urn:nbn:de:0072-395167.

Shahzad, F. (2014). »State-of-the-art survey on cloud computing security Challenges, approaches and solutions«. In: *Procedia Computer Science* 37, pp. 357–362.

Suchodoletz, D. von et al. (2014). »bwLehrpool – ein landesweiter Dienst für die Bereitstellung von PC-Pools in virtualisierter Umgebung für Lehre und Forschung«. In: *PIK – Praxis der Informationsverarbeitung und Kommunikation* 37.1, pp. 33–40. DOI: 10.1515/pik-2013-0046.

Tari, Z., X. Yi, U. S. Premarathne, P. Bertok and I. Khalil (2015). »Security and privacy in cloud computing: Vision, trends, and challenges«. In: *IEEE Cloud Computing* 2.2, pp. 30–38.

Wang, C., Q. Wang, K. Ren, N. Cao and W. Lou (2012). »Toward secure and dependable storage services in cloud computing«. In: *IEEE transactions on Services Computing* 5.2, pp. 220–232.

Wilkinson, M. D. et al. (2016). »The FAIR Guiding Principles for scientific data management and stewardship«. In: *Scientific data* 3. DOI: 10.1038/sdata.2016.18.

Zhao, F., C. Li and C. F. Liu (2014). »A cloud computing security solution based on fully homomorphic encryption«. In: *Advanced Communication Technology (ICACT)*. 16th International Conference on Advanced Communication Technology (ICACT). IEEE, pp. 485–488.

Zissis, D. and D. Lekkas (2012). »Addressing cloud computing security issues«. In: *Future Generation computer systems* 28.3, pp. 583–592.