Virtualized Research Environments on the bwForCluster NEMO

Michael Janczyk, Bernd Wiebelt, and Dirk von Suchodoletz
eScience Department, Computer Center, University of Freiburg, Freiburg, Germany
{michael.janczyk, bernd.wiebelt, dirk.von.suchodoletz}@rz.uni-freiburg.de

***Abstract*—The bwForCluster NEMO offers high performance computing resources to three quite different scientific communities (Elementary Particle Physics, Neuroscience and Microsystems Engineering) encompassing more than 200 individual researchers. To provide a broad range of software packages and deal with the individual requirements, the NEMO operators seek novel approaches to cluster operation [1]. Virtualized Research Environments (VREs) can help to both separate different software environments as well as the responsibilities for maintaining the software stack. Research groups become more independent from the base software environment defined by the cluster operators. Operating VREs brings advantages like scientific reproducibility, but may introduce caveats like lost cycles or the need for layered job scheduling. VREs might open advanced possibilities as e.g. job migration or checkpointing.**

***Index Terms*—Virtualized Research Environments, VRE, HPC, Virtualization, ViCE, bwHPC, bwHPC-C5**

I. Virtualized Research Environments

The author of [3] defines Virtual Research Environments to "[...] comprise digital infrastructure and services which enable research to take place. [...] A VRE is best viewed as a framework into which tools, services and resources can be plugged. VREs are part of infrastructure, albeit in digital form, rather than a free-standing product." While this more than a decade old paper emphasizes the general shared electronic infrastructure aspect, the term Virtualized Research Environment shifts the focus more to the technological viewpoint. Virtualization became a mainstream technology over the last decade as it allows the operators to host more than one operating system on a single server and to strictly separate users of software environments. While widespread in computer center operation, virtualization is still seen skeptically in the field of scientific computing and thus seldom applied in HPC.

The filesystem of a VRE is a container presented as a single file (or a couple of files linked together to present the container). From the operator's perspective this container is seen as a black box requiring no involvement or efforts like updates of the operating system or the provisioning of software packages of a certain version. From the researcher's perspective the VRE is an individual (virtual) node where everything from the hardware level – at least to a certain degree like CPU or RAM – up to the operating system, applications and configurations can be controlled solely by the research groups. Individual research groups become independent of the software and update strategy of the cluster operator. The caveat is that the old HPC performance mantra "every cycle counts" has to be sacrificed to a certain extent [5]. However, the advantages of this approach makes HPC attractive for broader scientific communities in the first place.

A. Reproducible Science

Digital research data management requires the conservation and publication of raw data as well as computed results. However, this is only half the story. It is equally important to preserve the computational tool chain used in the acquisition of new scientific insights. In the classic HPC approach, digital research environments are tightly coupled to the hardware and base operating system and therefore no longer usable when the HPC system is decommissioned. Instead, a lot of effort has to be put into reestablishing similar research environments on the successor system. By applying well-known techniques used in Cloud computing to HPC, deploying digital software environments to new HPC systems becomes a straightforward, simple and reproducible procedure. Even better, these *virtualized* digital research environments can be conserved in the state they were in when the actual scientific research was conducted. Establishing such certified research environments (CREs) – tested, certified and immutable software stacks – could be the next logical step. A CRE could thus help to complement electronic lab books.

B. ViCE Project on VREs

The eResearch initiative on Virtualized Research Environments (VREs) sponsored by the state of Baden-Württemberg provided the perfect framework to bring together both the infrastructure providers like the computer center of the University of Freiburg and research communities (e.g. Elementary Particle Physics from Karlsruhe and Freiburg) in the ViCE project. The research groups have various demands for computing in scientific workflows which are tightly coupled to well defined software environments. The project helped the involved parties to understand each others requirements to utilize virtualization and containerization technologies and to separate the software-hardware stack to foster more independence and flexibility for both of the parties. In daily operations, VREs help to disentangle the responsibilities regarding the digital research environment required for a certain task. The computer center focuses on the provisioning of a scalable hardware base including all necessary components to run virtualization and containerization technologies like OpenStack or Singularity.

II. Visions and Goals

In the classic HPC approach, research environments are meticulously handcrafted to squeeze out every last cycle count of the compute resources. How this is achieved is often hidden deep inside software module definitions created by software administrators or hidden inside source code modifications done by the user. It is easy to lose track of all these changes, especially in case there has been no agreed upon standard procedure put into place to record and conserve them.

When we look at software deployment on individual workstations or notebooks, the situation becomes even more inextricable. Hardware and software installation merge and converge into a monolithic masterpiece, fully understood solely by its creator – if at all. In any case, once the creator loses interest, his creation is doomed.

In contrast, a successful strategy in Cloud computing has been the orchestration of software environment deployment. The scientists create or modify recipes to generate Virtualized Research Environments tailored to their scientific workflows. These can be shared and improved upon by co-workers and collaborators. To this end, software packages (source or binary) are requested from well-defined and permanent repositories. VRE images (templates) get built on dedicated VRE build hosts. These VRE build hosts can be run as virtual machines on the user's workstation or in a Cloud service. After that, VRE instances can be deployed in the needed number on the Cloud system. This workflow is demonstrated in Fig. 1.

Ideally, these VRE instances should be agnostic of the environment they are started in. So they should be runnable in either Cloud or HPC environments. In the case of HPC, VRE instances should be considered as "just another job".

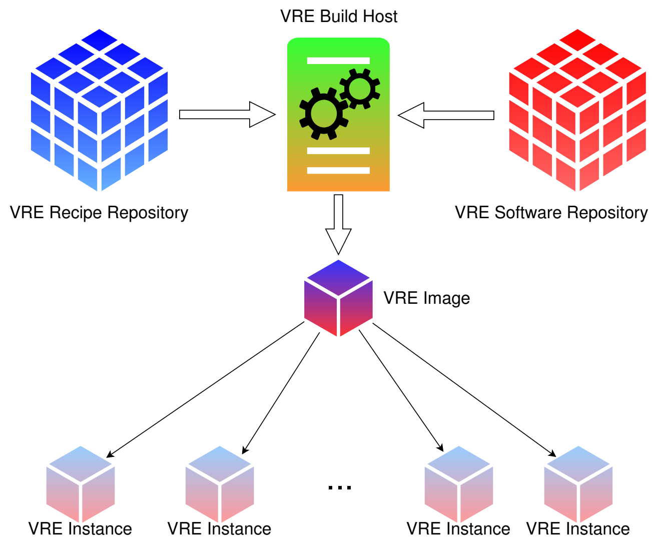

Fig. 1. Recipe and software repositories for automatic building of reproducible VREs.

III. Running VRE Instances on NEMO

On a HPC cluster users don't start their computations on the worker nodes themselves. This is done by the scheduler, who decides which job to put on which worker node. It can employ different strategies to do so and can take priorities and different fairness strategies into account. Compute Clouds use scheduling as well. Usually a set of system flavors – pre-configured virtual hardware sets – is given, which describes the resources one can get in the Cloud. In HPC it is custom that users get exactly the resources they asked for. If requirements can not be satisfied instantly, the job will have to wait in a queue. In Clouds the hardware resources usually are overbooked so that queues are processed within minutes and waiting time is minimized to the start of a machine. Commercial Clouds additionally steer their queues by monetary incentives.

Given these two system variants, HPC scheduling on the one side and Cloud scheduling on the other, competing for hardware resources in the cluster it is necessary to orchestrate both systems through a mediator. This mediator is called VRE service. It runs as a service daemon and waits for requests from users and administrators and translates it to the OpenStack API. In this hybrid approach one needs to define a primary scheduler who controls which jobs should run on which worker node and instruct the other scheduler to run within the boundaries of the other scheduler, especially respecting the scheduling decision of the other. The problem is, that neither the HPC scheduler, in our case the combination of the workload manager Moab and the resource manager Torque, nor the Cloud scheduler are aware of each other and no APIs exist on both sides to integrate a different scheduler. Since we provide a hybrid Cloud use case on a HPC cluster and users are familiar with submitting jobs from the console and Moab has the more sophisticated scheduling features compared to OpenStack we define Moab as the default scheduler. The OpenStack scheduler is still running but is told where to start its virtual machines. This way we do not need to integrate the Moab scheduler into OpenStack and that way this should work with every HPC / HTC scheduler on the market.

Starting virtual machines through the OpenStack dashboard is still allowed but only in a special service environment which does not interfere with the worker nodes of the HPC cluster. The VRE service is monitoring all virtual machines started and destroys every machine which is started outside of the service environment when no corresponding Moab job ist found.

Starting Virtualized Research Environments on the worker nodes is done through the Moab scheduler by submitting special tags as variables. The scheduler queues and routes this VRE job as a standard HPC job respecting policies like fair-share, usage limits and the requested resources like memory, CPU and wall-time. Once the job is started on the node the torque client (mother superior) filters it for the special VRE tags and submits this request to the VRE service. The VRE service then translates this information to the corresponding tuple of OpenStack project, VRE name, flavor for resources and binds it to the corresponding node. This information is then send to the OpenStack service which then starts the VRE (a virtual machine) on the pre-selected worker node. If the Moab job ends or is canceled by the system or user the mother superior sends this information to the VRE service which then destroys the instance. During the lifetime of the job the VRE service is monitoring the virtual machine and reports back utilization when requested by users or services. No direct interaction with the OpenStack service is necessary by users or administrators. The principle can be seen in Fig. 2.

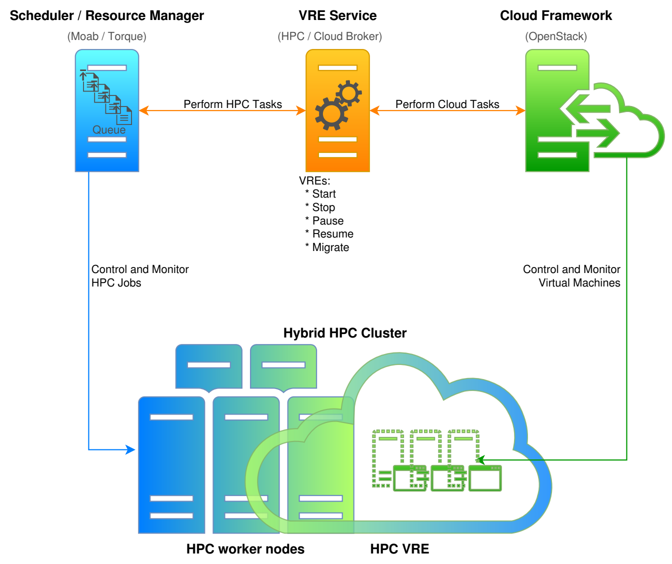

Fig. 2. VRE service acts as broker between the two schedulers.

The helper tool `cloud-init`, which is used to customize virtual machine images on boot, is used to inject scripts to execute within the VRE, if provided. The job description file, which defines the resources needed and the commands to execute during the job's run-time, and usually is used on compute clusters is replaced by a simple wait script to ensure that the HPC job is not ended before the VRE job has finished.

The VRE service is developed in a cooperation between the HPC team in Freiburg and Adaptive Computing, the company behind Moab. The concept is still in development, so future results can differ from this description. The following lines show an example for submitting VREs on NEMO:

```
msub -l nodes=1:ppn=20 \
     -l pmem=6GB \
     -l walltime=1:00:00:00 \
     -l var=vmscript:$script+vmimage: ↩
        $image+vmproject:$project+ ↩
        vmsshkeyname:$key \
     job_dummy.sh
```

IV. Bioinformatics and High Energy Physics VREs on NEMO

A typical concern for many scientific workflows, especially in (geographically) distributed large projects, is the strict requirement for the software stack involved [1].

Therefore the huge CERN projects like ATLAS and CMS usually are very conservative with changes in their environment and usually test future environments over months or even years. When this paper was written the ATLAS and CMS project still used Scientific Linux 6.x as operating system while the bwForCluster NEMO used the newer operating system CentOS 7.4. The complete software and service stack of the High Energy Physics (HEP) is tailored around this Scientific Linux 6. Driven by the fact that Elementary Particle Physics is one of the research fields on the bwForCluster NEMO (the others being Neuroscience and Microsystems Engineering) a solution was needed to provide the HEP research environments on the cluster. By using VREs ATLAS and CMS [6] users have now the possibility to run their jobs in their standard HEP research environments and it is guaranteed, that the operating system and the software stack is compatible with the HEP environments used on bare-metal clusters.

The Bioinformatics department uses the the Galaxy Portal for their computations. To support this environment on NEMO a VRE including Galaxy has been created and the integration into the NEMO cluster is in development currently.

V. Advanced Concepts

Using virtualization inside a HPC system opens up possibilities for several interesting features. While their implementation requires tighter integration between HPC scheduler and virtualization framework as shown in Fig. 2, they could solve several classic problems with HPC systems, especially those designated for novice HPC users.

Using the virtualization features "snapshot and migration" for a virtualized cluster like e.g. described in [4], [5], would enable operators and users to stop running processes and move them to a different node in the cluster where they could be resumed. For an HPC system, this would be practical for two use cases. The first one concerns long running monolithic jobs. These are, for very practical reasons, non-favored jobs in HPC environments, assuming they are permitted in the first place. However, the costs to adapt a particular workflow based on such monolithic tasks to a HPC system, e.g. by parallelizing and partitioning it manually, may sometimes exceed the practical use of the resulting solution. If the monolithic job could automatically be stopped, check-pointed and resumed at regular intervals, this might very well constitute a more economic procedure. In the second use case, if there is a mix of pleasingly parallel high throughput jobs (using only single cores or nodes) and massively parallel high performance jobs (using several nodes), the second class of jobs should be concentrated on nodes that share optimal high performance network communication paths. Typically this is accomplished by high investments in the network topology or sophisticated tuning of the job queue. However, if jobs could be moved around the physical machines (i.e. "de-fragmented"), optimal high performance network communication paths can be guaranteed by concentrating massively parallel jobs on the same or adjacent high performance network switches.

VI. Outlook

There still exist a couple of limitations on the presented concepts and workflows. The use cases considered feature embarrassingly parallelized workloads which do not require interaction between individual cluster nodes. This simplifies the setup and operation of virtual machines as the high bandwidth node interconnect could be ignored in this scenario. Singularity might overcome that limitation in the future but would make the application of the advanced concepts more complex as explained in [4]. A further challenge arises from the provisioning of the data in the form of parallelized high performance storage. If users become root in their VREs than the traditional means of privilege separation will not work any more. A similar issue arises from the heightened mobility of VREs. While in static cluster configurations IP based security with its limitations made sense, this becomes more challenging when VREs get involved. Depending on the way virtualization or containerization is implemented, the HPC scheduling setup has to be aware of the more dynamic nature of resources.

Pre- and post-processing in many workflows often profit from interactive intervention instead of batch-driven automatic processing. Here, VREs could help to use the same working environment for Cloud (pre- and post-processing) and HPC systems (main computational task). Beside the organizational and administrative benefits VREs might allow to gain more flexibility through e.g. job checkpoint-restart or job migration. Because of the containerization of VREs these could get extended to Certified Research Environments in the future.

Acknowledgment

The authors acknowledge support by the state of Baden-Württemberg through bwHPC and the ViCE project as well as support by the German Research Foundation (DFG) through grant no INST 39/963-1 FUGG.

References

[1] K. Meier, G. Fleig, T. Hauth, M. Janczyk, G. Quast, D. von Suchodoletz, and B. Wiebelt, "Dynamic provisioning of a HEP computing infrastructure on a shared hybrid HPC system," 17th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2016), Valparaiso Chile, Proceedings, p. 18–22, January 2016

[2] D. von Suchodoletz, B. Wiebelt, K. Meier, and M. Janczyk, "Flexible HPC: bwForCluster NEMO," Proceedings of the 3rd bwHPC-Symposium: Heidelberg 2016, editors: S. Richling, M. Baumann, and V. Heuveline, doi: 10.11588/heibooks.308.418, heiBOOKS, 2017

[3] M. Fraser, "Virtual research environments: overview and activity," num. 44, url: http://www.ariadne.ac.uk/issue44/fraser, Ariadne, 2005

[4] P. Anedda, S. Leo, S. Manca, M. Gaggero, and G. Zanetti, "Suspending, migrating and resuming HPC virtual clusters," Future Generation Computer Systems, vol. 26, num. 8, p. 1063–1072, Elsevier, 2010

[5] A. Younge, R. Henschel, J. Brown, G. von Laszewski, J. Qiu, and G. Fox, "Analysis of virtualization technologies for high performance computing environments," 2011 IEEE International Conference on Cloud Computing (CLOUD), p. 9–16, IEEE, 2011

[6] T. Hauth, G. Quast, M. Kunze, V. Büge, A. Scheurer, and C. Baun, "Dynamic extensions of batch systems with cloud resources," Journal of Physics: Conference Series, vol. 331, num. 6, p. 062034, IOP Publishing, 2011