Noname manuscript No.
(will be inserted by the editor)

Dynamic Virtualized Deployment of Particle Physics Environments on a High Performance Computing Cluster

Felix Bührer · Frank Fischer · Georg Fleig · Anton Gamel · Manuel Giffels · Thomas Hauth · Michael Janczyk · Konrad Meier · Günter Quast · Benoît Roland · Markus Schumacher · Ulrike Schnoor · Dirk von Suchodoletz · Bernd Wiebelt

Received: date / Accepted: date

F. Bührer
Universität Freiburg, Physikalisches Institut, Hermann-Herder-Str. 3, 79104 Freiburg, Germany

F. Fischer
Karlsruher Institut für Technologie, Institut für Experimentelle Teilchenphysik, Wolfgang-Gaede-Str. 1, 76131 Karlsruhe, Germany

G. Fleig
Karlsruher Institut für Technologie, Institut für Experimentelle Teilchenphysik, Wolfgang-Gaede-Str. 1, 76131 Karlsruhe, Germany

A. Gamel
Universität Freiburg, Physikalisches Institut, Hermann-Herder-Str. 3, 79104 Freiburg, Germany
Universität Freiburg, Rechenzentrum, Hermann-Herder-Str.10, 79104 Freiburg, Germany

M. Giffels
Karlsruher Institut für Technologie, Institut für Experimentelle Teilchenphysik, Wolfgang-Gaede-Str. 1, 76131 Karlsruhe, Germany

T. Hauth
Karlsruher Institut für Technologie, Institut für Experimentelle Teilchenphysik, Wolfgang-Gaede-Str. 1, 76131 Karlsruhe, Germany

M. Janczyk
Universität Freiburg, Rechenzentrum, Hermann-Herder-Str. 10, 79104 Freiburg, Germany

K. Meier
Universität Freiburg, Rechenzentrum, Hermann-Herder-Str. 10, 79104 Freiburg, Germany

G. Quast
Karlsruher Institut für Technologie, Institut für Experimentelle Teilchenphysik, Wolfgang-Gaede-Str. 1, 76131 Karlsruhe, Germany

B. Roland
Universität Freiburg, Physikalisches Institut, Hermann-Herder-Str. 3, 79104 Freiburg, Germany

M. Schumacher
Universität Freiburg, Physikalisches Institut, Hermann-Herder-Str. 3, 79104 Freiburg, Germany

U. Schnoor
Universität Freiburg, Physikalisches Institut, Hermann-Herder-Str. 3, 79104 Freiburg, Germany
E-mail: ulrike.schnoor@cern.ch
Now at CERN, CH-1211 Geneva 23, Switzerland

D. von Suchodoletz
Universität Freiburg, Rechenzentrum, Hermann-Herder-Str. 10, 79104 Freiburg, Germany

B. Wiebelt
Universität Freiburg, Rechenzentrum, Hermann-Herder-Str. 10, 79104 Freiburg, Germany

Abstract The NEMO High Performance Computing Cluster at the University of Freiburg has been made available to researchers of the ATLAS and CMS experiments. Users access the cluster from external machines connected to the World-wide LHC Computing Grid (WLCG). This paper describes how the full software environment of the WLCG is provided in a virtual machine image. The interplay between the schedulers for NEMO and for the external clusters is coordinated through the `ROCED` service. A cloud computing infrastructure is deployed at NEMO to orchestrate the simultaneous usage by bare metal and virtualized jobs. Through the setup, resources are provided to users in a transparent, automatized, and on-demand way. The performance of the virtualized environment has been

Keywords Virtualization · Particle Physics · Grid Computing · Benchmarks

1 Introduction

Particle physics experiments at the Large Hadron Collider (LHC) need a great quantity of computing resources for data processing, simulation, and analysis.

This demand will be growing with the upcoming High-Luminosity upgrade of the LHC [1]. To help fulfill this requirement, High Performance Computing (HPC) resources provided by research institutions can be useful supplements to the existing World-wide LHC Computing Grid (WLCG) resources allocated by the collaborations.

This paper presents the concepts and implementation of providing a HPC resource, the shared research cluster NEMO at the University of Freiburg, to ATLAS and CMS users accessing external clusters connected to the WLCG with the purpose of accomodate data production as well as data analysis on the HPC host system. The HPC cluster NEMO at the University of Freiburg is deploying an `OpenStack` [2] instance to handle the virtual machines. The challenge is in provisioning, setup, scheduling, and decommissioning the virtual research environments (VRE) dynamically and according to demand. For this purpose, the schedulers on NEMO and on the external resources are connected through the `ROCED` service [3].

A VRE in the context of this paper is a complete software stack as it would be installed on a compute cluster fitted to the demands of ATLAS or CMS workloads.

2 Virtualization infrastructure

Hardware virtualization has become mainstream technology over the last decade as it allows both to host more than one operating system on a single server and to strictly separate users of software environments. Hardware and software stacks are decoupled and therefore complete software environment can be migrated across hardware boundaries. While widespread in computer center operation this technique is rarely applied in HPC.

2.1 Computing at the University of Freiburg

The computer center at the University of Freiburg provides medium scaled research infrastructures like cloud, storage, and especially HPC services adapted to the needs of various scientific communities. Significant standardization in hardware and software is necessary for the operation of compute systems comprised of more than 1000 individual nodes combined with a small group of administrators.

The level of granularity of the software stack provided is not fine enough to directly support the requirements of world-wide efforts like the ATLAS or CMS experiments. Therefore, novel approaches are necessary to ensure optimal use of the system and to open the cluster to as many different use-cases as possible without increasing the operational effort. Transferring expertise from the operation of the established local private cloud, the use of `OpenStack` as a cloud platform has been identified as a suitable solution for NEMO. This approach provides a user defined software deployment in addition to the existing software module system. The resulting challenges range from the automated creation of suitable virtual machines to their on-demand deployment and scheduling.

2.2 Research Cluster NEMO

The research cluster NEMO is a cluster for state-wide research in the scientific fields Elementary Particle Physics, Neuroscience and Microsystems Engineering. It started its operation on the 1st of August 2016 and consists currently of 900 nodes with 20 physical cores and 128 GiB of RAM each. Omni-Path [4] spans a high speed low latency network of 100 Gbit/s between nodes. The parallel storage is based on BeeGFS [5] with 768 TB capacity.

2.3 Separation of software environments

The file system of a virtual machine or VRE is a disk image presented as a single file. From the computer center's perspective this image is 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 whose operating system, applications and configurations as well as certain hardware-level parameters, e.g. CPU and RAM, can be configured fully autonomously by the researcher within agreed upon limits.

To increase the flexibility in hosted software environments, the standard bare metal operation of NEMO is extended with an installation of `OpenStack` components [6]. The NEMO cluster uses Adaptive's Workload Manager `Moab` [7] as a scheduler of compute jobs. `OpenStack` as well can schedule virtual machines on the same nodes and resources. To avoid conflicts, it is necessary to define the master scheduler which decides the job assignment to the worker nodes. Both `Moab` and `OpenStack` are unaware that another scheduler exists within the cluster and there is no API which enables them to communicate with each other. Since the majority of users still use the bare metal HPC cluster, `Moab` is deployed as the primary scheduler. It allows for detailed job description and offers sophisticated scheduling features like fair-share, priority-based scheduling, detailed limits, etc. OpenStack's task is to deploy the virtual machines, but Moab will initially start the VRE jobs and the VRE job will instruct OpenStack to start the virtual machine on the reserved resources with the required flavor, i.e. the resource definition in OpenStack.

When a VRE job is submitted to the NEMO cluster, Moab will first calculate the priority and the needed resources of the job and then inserts it into its queue. When the job is in line for execution and the requested resources are available, the job will start a script which then starts the VRE on the selected node within the resource boundaries. During the run-time of the VRE a monitoring script regularly checks if the VRE is still running and terminates the job when the VRE has ended. When the job ends, OpenStack gets a signal to terminate the virtual machine and the VRE job ends as well. Neither Moab nor OpenStack have access inside the VRE, so they cannot assess if the VRE is actually busy or idle. The software package ROCED (described in further detail in Section 4) has been introduced to solve this issue. It is used as a broker between different HPC schedulers, translating resources and monitoring usage inside the virtual machine, as well as starting and stopping VRE images on demand.

3 Generation of the VRE image

The VREs for ATLAS and CMS software environments consist in OpenStack containers in the format of compatible VM images. These images are provided in an automatized way allowing versioning and archiving of the environments captured in the images.

3.1 Packer combined with Puppet

A reasonable approach to generate the image is the open-source tool Packer [8], interfaced to the system configuration framework Puppet [10]. Packer allows to configure an image based on an ISO image file using a kickstart [9] file and flexible script-based configuration. It also provides an interface to Puppet making it particularly convenient if an existing Puppet role is to be used for the images. If the roles are defined according to the hostname of the machine as is conventional in Puppet with Hieradata, the hostname needs to be set in the scripts supplied to Packer. Propagation of certificates requires an initial manual start of a machine with the same hostname to allow handshake signing of the certificate from the Puppet server.

Packer's interface to Puppet allows a fully automated image generation with up-to-date and version-controlled configuration. At the end of the generation run, the image is automatically transferred to the OpenStack image server.

3.2 Image generation using the Oz toolkit

Another option to employ a fully-automated procedure is to use the OZ toolkit [11]. All requirements and configuration options of an image can be specified through a XML template file. The partitioning and installation process of the operating system is fully automated, as OZ will use the remote-control capabilities of the local hypervisor. After the installation of the operating system, additional libraries and configuration files can be installed. Once the image has been created, it is automatically compressed and uploaded to a remote cloud site. This technique allows to build images in a reproducible fashion, as all templated files are version controlled using git. Furthermore, existing template files are easy to adapt to new sites and experiment configurations.

4 Interfacing batch systems and virtual resources using ROCED

While HPC systems with support for virtualized research environments and commercial cloud providers offer the necessities to acquire computing and storage capacity by dynamic resource booking, the computing needs of high energy physics research groups additionally require workflow management systems capable of maintaining thousands of batch jobs. Some cloud providers, for example Amazon with AWS Batch [12], provide a service for workflow management, however these offerings are often limited to one specific cloud instance. To dynamically distribute batch jobs to multiple sites and manage machine life-time on specific sites, a combination of a highly-scalabe batch system and a virtual machine scheduler is desirable.

4.1 ROCED

Many capable batch systems exist today and they can be interfaced to virtualization providers using the cloud meta-scheduler ROCED (Responsive On-demand Cloud Enabled Deployment) which has been developed at the KIT since 2010 [3]. ROCED is written in a modular fashion in python and the interfaces to batch systems and cloud sites are implemented as so-called *Adapters*. This

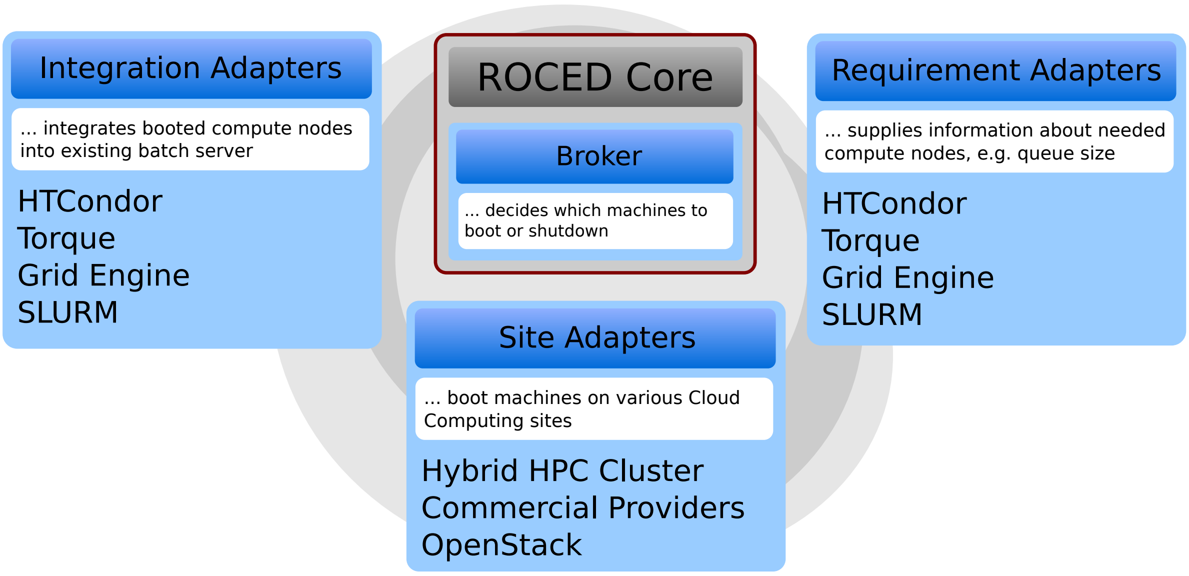

Fig. 1 Overview of the `ROCED` modular design. The `ROCED` Core contains the Broker which decides when and on which sites new virtual machines are booted. The Requirement Adapters report about the utilization and resource requirements of the attached batch systems. The Site Adapter is responsible to manage the lifetime of virtual machines on an cloud site and the Integration Adapter ensure that newly booted machines are integrated into the batch system.

makes `ROCED` independent of specific user groups or workflows. It provides a scheduling core which collects the current requirement of computing resources and decides if virtual machines need to be started or can be stopped. One or more Requirement Adapters report the current queue status of batch systems to the central scheduling core. Currently, Requirement Adapters are implemented for the Slurm, Torque/Moab, HTCondor and GridEngine batch systems. The Site Adapters allow `ROCED` to start, stop, and monitor virtual machines on multiple cloud sites. Implementations exist for Amazon EC2, OpenStack, OpenNebula and Moab-based virtualization at HPC centers. Special care has been put into the resilience of `ROCED`: it can automatically terminate non-responsive machines and restart virtual machines in case some machines have dropped out. This allows VM setups orchestrated by `ROCED` with thousands of virtual machines and many tens of thousands of jobs to run in production environments. The modular design of `ROCED` is shown in Fig. 1.

4.2 Using HTCondor as front-end scheduler

The open-source project `HTCondor` provides a workload management system which is highly configurable and modular [13]. Batch processing workflows can be submitted and are then forwarded by `HTCondor` to idle resources. `HTCondor` maintains a resource pool, which worker nodes in a local or remote cluster can join. Once `HTCondor` has verified the authenticity and features of the newly joined machines, computing jobs are automatically transferred. Special features are available to connect from within isolated network zones, e.g. via a Network Address Translation Portal, to the central `HTCondor` pool. The Connection Brokering (CCB) service [14] is especially valuable to connect virtual machines to the central pool. These features and the well-known ability of `HTCondor` to scale to O(100k) of parallel batch jobs makes `HTCondor` well suited as a workload management system for the use cases described in this paper.

The virtual machines spawned for the CMS user group of the KIT come with `startd` the `HTCondor` client pre-installed. This client is started after the machine has fully booted and connects to the central `HTCondor` pool at the KIT via a shared secret. Due to `HTCondor`'s dynamic design, new machines in the pool will automatically receive jobs and the transfer of the job configuration and meta-data files is handled via `HTCondor`'s internal file transfer systems.

4.3 Using SLURM as front-end scheduler

Alternatively to the approach described previously, the open-source workload managing system `Slurm` [15] has been interfaced into `ROCED` by the ATLAS group at University of Freiburg. While `Slurm` provides a built-in functionality for dynamic startup of resources in the *Slurm Elastic Computing* module [16], this has been

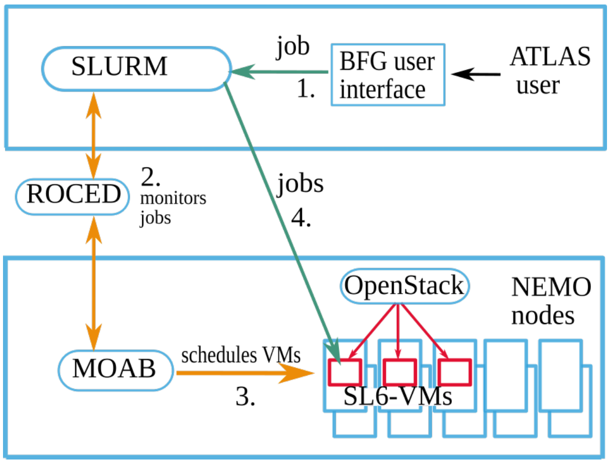

Fig. 2 Implementation of `ROCED` with `Slurm` on the BFG cluster used by ATLAS researchers.

found to be unsuitable for resources which are not expected to be available within a fixed time period, in this case due to the presence of a queue in the host system which may postpone the start of a resource by a significant, varying period. In addition the transfer of information, such as error states, from one scheduler to the other, and therefore to the user, is very limited. Therefore, `ROCED` has been chosen as the interface between the `Moab` scheduler on the host system and the `Slurm` scheduler on the submission side.

The scheduling system is illustrated in Fig. 2. For `Slurm`, it is necessary that each potential virtual machine is registered in the configuration at the time of start of the `Slurm` server as well as the client. `Slurm` configurations also need to be in agreement between server and client. Therefore, a range of hostnames is registered in the configuration in a way that is mapped to potential IP addresses of virtual machines. These virtual machines have a fixed number of CPUs and memory assigned and are registered under a certain `Slurm` partition. When a job is submitted to this partition and no other resource is available, information from the `Slurm` `squeue` and `sinfo` commands is requested and parsed for the required information.

Since the ATLAS Freiburg group comprises three sub-groups, each mapped to a different production account on NEMO, special care is taken to avoid interference of resources used by another account to ensure fair share on NEMO, while allowing jobs from one group to occupy otherwise idle resources of another group.

`ROCED` determines the amount of virtual machines to be started and sends the corresponding VRE job submission commands to `Moab`. After the virtual machine has booted, the hostname is set to the IP dependent name which is known to the `Slurm` configuration. A `cron` job executes several sanity checks on the system. Upon successful execution of these tests, the `Slurm` client running in the VM starts accepting the queued jobs. After completion of the jobs and a certain period of receiving no new jobs from the queue, the `Slurm` client in the machine drains itself and the machine shuts itself down. The IP address as well as the corresponding hostname in `Slurm` are released and can be reused by future VREs.

5 Analysis of performance and usage

This approach has been implemented and put into production by the research groups at the University of Freiburg (Institute of Physics) and the Karlsruhe Institute of Technology (Institute of Experimental Particle Physics). The following section presents the statistical analysis of the performance of the virtualized setup both in terms of CPU benchmarks and usage statistics.

5.1 Benchmarks

Alongside the legacy HEP-SPEC06 (HS06) benchmark [17], the performance of the compute resources is furthermore evaluated with the ATLAS Kit Validation KV [20], a fast benchmark developed to provide real-time information of the WLCG performance and available in the CERN benchmark suite [18]. The KV benchmark is making use of the simulation toolkit GEANT4 [21] to simulate the interactions of single muon events in the detector of the ATLAS experiment and provides as ouput the number of events produced per second. As our primary target is to measure performances of CPUs in the context of High Energy Physics applications, the KV benchmark constitutes a realistic workload.

To assess the impact of the virtualization, the performance of the same hardware configuration (20 cores Intel Xeon E5-2630 CPUs) has been determined either deployed via the standard bare metal operation on the NEMO cluster (NEMO bare metal) and on the ATLAS Tier-3 center in Freiburg (ATLAS Tier-3 bare metal), or as virtual machines on the NEMO cluster (NEMO VM). On the ATLAS Tier-3 bare metal and on the virtual machines running on the NEMO cluster, hyper-threading (HT) technology is activated. Both are using Scientific Linux 6 [22] as operating system. The NEMO bare metal has no HT activated due to the more general use case of the system, and uses CentOS7 as operating system [23]. The scores of the HEP-SPEC06 and KV benchmarks have been determined for these three configurations as a function of the number of cores actually used by the benchmarking processes. This number ranges from 2 to 40 for the ATLAS Tier-3 bare metal and for the NEMO VM, for which HT is enabled, and from 2 to 20 for the NEMO bare metal, for which HT is not implemented. The benchmarks have been run 20 times for each core multiplicity value, and the means and standard deviations of the corresponding distributions have been extracted.

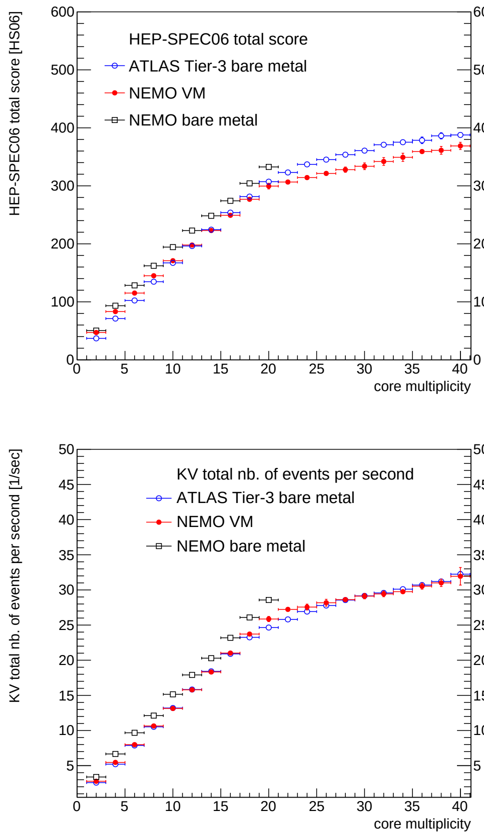

Fig. 3 Total score as a function of the core multiplicity for the HEP-SPEC06 (top) and KV (bottom) benchmarks for the ATLAS Tier-3 bare metal (blue open circles), the NEMO VMs (red full circles) and the NEMO bare metal (black open squares). The data points represent the average values of the benchmarks for each core multiplicity, and the vertical bars show the associated standard deviations.

The HEP-SPEC06 and KV results are presented in Figure 3 for the three configurations considered. The total scores of the two benchmarks are increasing until the maximum number of physical cores has been reached, and are characterized by a flattening increase afterwards. The scores of the virtual machines running on the NEMO cluster are only slightly lower than those obtained for the NEMO bare metal, and the loss of performance due to the virtualization does not exceed 10%. For the VMs running on the NEMO cluster and

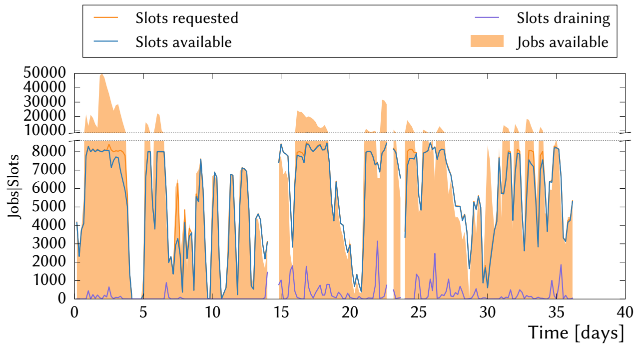

Fig. 4 Utilization of the shared HPC system by booted virtual machines. Up to 9000 virtual cores were in use at peak times. The fluctuations in the utilization reflects the patterns of the submission of jobs by our institute users. The number of draining slots displays the amount of job slots still processing jobs while the rest of the node's slot are already empty.

the ATLAS Tier-3 bare metal, the interplay between the virtualization and the different operating systems leads to very similar scores for the two configurations, particularly for the KV benchmark, and the loss of performance is smaller than 10% as well.

5.2 Usage statistics

Fig. 4 shows the utilization of virtual machines which were orchestrated by `ROCED` depending on the resource demands of the users of the KIT group. At peak times, up to 9000 virtual cores were filled with user jobs, consuming more than a half of the initial 16000 NEMO cores.

The usage of the hybrid cluster model is presented in Fig. 5. The diagram shows the shared usage of NEMO's cluster nodes running either bare-metal or virtualized jobs. The part of the cluster which runs virtualized jobs or VREs changes dynamically from job to job, since the VREs are started by a standard bare-metal job.

At the beginning the cluster was only containing the operating system and some basic development tools, scientific software was added after the cluster was already in production mode. Since the VRE for the CMS project was already available when the NEMO cluster started, it could already use the whole cluster while other groups still had to wait for the required scientific software to be deployed on the cluster. This explains the high usage by VREs in the first months of operation. With more and more software being available for bare-metal usage the amount of VRE jobs decreased. This figure is only an estimate because VRE projects are not forced to use VREs and therefore could run bare-metal jobs as well.

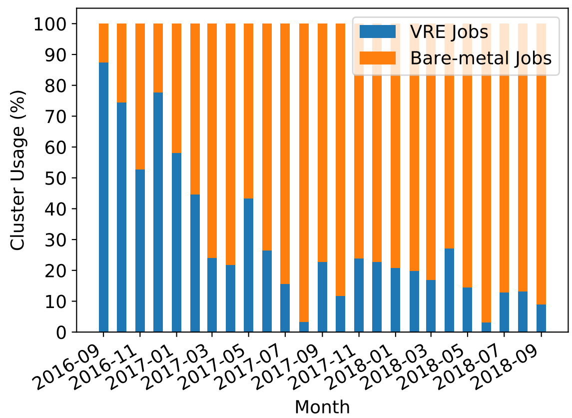

Fig. 5 Usage of the NEMO cluster in the time between September 2016 to June 2018. The blue bars indicate the usage by jobs running directly in the hosts' operating system, while the orange bars are jobs running in virtual machines.

6 Conclusions and Outlook

A system for the dynamic, on-demand provisioning of virtual machines to run jobs in a high energy physics context on an external, not dedicated resource as realized at the HPC cluster NEMO at the University of Freiburg has been described. An interface between the schedulers of the host system and the external system from which requests are sent is needed to monitor and steer jobs in a scalable way. This is implemented in the `ROCED` package which is deployed for the described use-cases. This approach can be generalized to other platforms and possibly also other forms of virtualized environments (e.g. containers).

The CPU performance and usage of the setup have been analyzed. The expected performance loss due to the virtualization has been found to be sufficiently small to be compensated by the added flexibility and other benefits of this setup.

A possible extension of such a virtualized setup is the provisioning of functionalities for snapshots and migration of jobs. This would facilitate the efficient integration of long-running monolithic jobs into HPC clusters.

Acknowledgements This research is supported by the Ministry of Science, Research and the Arts Baden-Württemberg through the bwHPC grant and by the German Research Foundation (DFG) through grant no INST 39/963-1 FUGG for the bwForCluster NEMO. The work of F.B. was supported by the Virtual Open Science Collaboration Environment (ViCE) project MWK 34-7547.221 funded by the Ministry of Science, Research and the Arts Baden-Württemberg. The work of U.S. was supported by the Bundesministerium für Bildung und Forschung in the project 05H15VFCA1 "Higgs-Physik mit dem und Grid-Computing für das ATLAS-Experiment am LHC".

References

1. ATLAS Public results `https://twiki.cern.ch/twiki/pub/AtlasPublic/ComputingandSoftwarePublicResults/diskHLLHC.pdf`, accessed 2018-09-19
2. OpenStack Open Source Cloud Computing Software `https://www.openstack.org/`, accessed 2018-07-03
3. `ROCED` Cloud Meta-Scheduler project website `https://github.com/roced-scheduler/ROCED`, accessed 2018-07-03
4. Omni-Path: "Intel Architects High Performance Computing System Designs to Bring Power of Supercomputing Mainstream", `https://newsroom.intel.com/news-releases/intel-architects-high-performance-computing-system-designs-to-bring-power-of-supercomputing-mainstream`, Intel. 16 November 2015, accessed 2018-09-20
5. BeeGFS Parallel Cluster File system: `https://www.beegfs.io/content/`, accessed 2018-09-20
6. Dirk von Suchodoletz, Bernd Wiebelt, Konrad Meier, Michael Janczyk, Flexible HPC: bwForCluster NEMO, Proceedings of the 3rd bwHPC-Symposium: Heidelberg 2016,
7. Adaptive Computing Moab `http://www.adaptivecomputing.com/moab-hpc-basic-edition/`, accessed 2018-07-03
8. Packer: tool for creating machine and container images for multiple platforms from a single source configuration. `https://www.packer.io/`, accessed 2018-07-03
9. `https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/installation_guide/ch-kickstart2`, accessed 2018-07-03
10. Puppet Enterprise. "IT automation for cloud, security, and DevOps." `https://puppet.com/`, accessed 2018-07-03
11. Oz image generation toolkit `https://github.com/clalancette/oz`, accessed 2018-07-03
12. Amazon AWS Batch `https://aws.amazon.com/batch/`, accessed 2018-07-03
13. HTCondor workload manager `https://research.cs.wisc.edu/htcondor/`, accessed 2018-07-03
14. HTCondor Connection Brokering `http://research.cs.wisc.edu/htcondor/manual/v8.6/3_9Networking_includes.html`, accessed 2018-07-03
15. Slurm `https://slurm.schedmd.com`, accessed 2018-07-03
16. Slurm Elastic Computing `https://slurm.schedmd.com/elastic_computing.html`, accessed 2018-07-03
17. HEPiX Benchmarking Working Group: `https://twiki.cern.ch/twiki/bin/view/FIOgroup/TsiBenchHEPSPEC`, accessed 2018-01-29
18. M. Alef *et al.*, "Benchmarking cloud resources for HEP", J. Phys. Conf. Ser. **898** (2017) no.9, 092056. doi:10.1088/1742-6596/898/9/092056
19. Graciani, Ricardo and Andrew McNab, Dirac benchmark 2012, `https://gitlab.cern.ch/mcnab/dirac-benchmark/tree/master`
20. A. De Salvo and F. Brasolin, "Benchmarking the ATLAS software through the kit validation engine", J. Phys. Conf. Ser. **219** (2010) 042037. doi:10.1088/1742-6596/219/4/042037
21. S. Agostinelli *et al.* [GEANT4 Collaboration], "GEANT4: A Simulation toolkit", Nucl. Instrum. Meth. A **506** (2003) 250. doi:10.1016/S0168-9002(03)01368-8
22. Fermilab and CERN, "Scientific Linux 6", `http://www.scientificlinux.org/`

23. The CentOS Project, “CentOS Linux 7”, `https://www.centos.org/`
24. P. Nason, JHEP 0411 (2004) 040, hep-ph/0409146; S. Frixione, P. Nason and C. Oleari, JHEP 0711 (2007) 070, arXiv:0709.2092; S. Alioli, P. Nason, C. Oleari and E. Re, JHEP 1006 (2010) 043, arXiv:1002.2581
25. T. Sjstrand et al: An Introduction to PYTHIA 8.2. Comput. Phys. Commun. 191 (2015) 159-177. DOI:10.1016/j.cpc.2015.01.024”. arXiv hep-ph 1410.3012