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

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

Felix Bührer · Anton Gamel · Michael Janczyk · Benoit Roland · Markus Schumacher · Ulrike Schnoor · Bernd Wiebelt · Thomas Hauth

Received: date / Accepted: date

Abstract Particle physics experiments at the Large Hadron Collider (LHC) need a great quantity of computing resources for data processing, simulation, and analysis. 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. At the University of Freiburg, the shared research cluster NEMO has been made available to ATLAS and CMS users accessing it from external machines connected to the WLCG. To this effect, the full software environment corresponding to a WLCG center is provided in a virtual machine image. The interplay between the schedulers for NEMO and for the external cluster is ensured through the ROCED service. An OpenStack 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 evaluated for particle physics applications.

Keywords Virtualization · Particle Physics · Grid Computing

1 Introduction

This paper presents the concepts and implementation of providing a HPC resource to ATLAS and CMS users accessing external clusters connected to the World-wide computing grid (WLCG) with the purpose of running data production as well as data analysis on the HPC host system. For this purpose, the HPC cluster NEMO at the University of Freiburg is deploying an OpenStack 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 [2].

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.

U. Schnoor
CERN
E-mail: ulrike.schnoor@cern.ch

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. Hard and software stacks are decoupled and therefore complete software environment can be migrated easily. 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 to cater 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 with a small group of administrators.

The level of granularity of the software stack provided is not fine enough to directly support the quite special requirements of world-wide efforts like the ATLAS or CMS experiments. To utilize the system optimally and to open the cluster to as many different needs as possible without increasing the person power required for operation, novel approaches to the utilization of the installed hardware are necessary. Transferring expertise from the operation of the established local private cloud, the use of OpenStack as a cloud platform has been identified in cooperation with the ViCE project [22] as a suitable solution for NEMO. This approach provides a more flexible software deployment in addition to the existing software module system. This produced a couple of implications ranging from challenges in the automated creation of suitable virtual machines, their on-demand deployment and the scheduling on the cluster and virtual resources level.

2.2 Research Cluster NEMO

The research cluster "bwForCluster 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 with then 748 nodes having 20 physical cores and 128 GiB of RAM each. Omni-Path spans a high speed low latency network of 100 Gbit/s between nodes. The parallel storage is BeeGFS with 576 TB capacity. In October 2017 the NEMO cluster was extended by research groups and offers currently 900 nodes and a total capacity of 768 TB of parallel storage.

2.3 Separation of software environments

The file system of a virtual machine (or virtualized research environment in the described use case) is a disk image presented as a single file. From the computer center's perspective this image 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 researchers 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 autonomously by the research groups.

To allow more flexible software environments the standard bare metal operation of NEMO is extended with a parallel installation of OpenStack components [12]. The NEMO cluster uses Adaptive's Moab Workload Manager [3] as a scheduler of compute jobs. Since OpenStack tries to schedule the virtual machines on the same nodes and resources itself, it is necessary to define a primary or master scheduler, who controls which jobs should run on which worker node. Moab and Openstack are unaware that there exists another scheduler within the cluster and no API exists so that both can communicate with each other. Since most 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 and much more. Openstack will still schedule 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 and which flavor to use (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 insert 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 checks if the VRE is still running and exits the job if the VRE ends. When the job ends, OpenStack gets a signal to terminate the virtual machine and the VRE job ends as well. Neither Moab nor OpenStack can look into the VRE and so it cannot assess if it is actually active or idle. To solve this issue, a customized glue component called ROCED has been introduced (described in further detail in Section 4). It is used as a broker between different HPC schedulers. It translates resources and monitors usage inside the virtual machine and starting and stopping VRE images on demand.

3 Generation of the image

The VREs for ATLAS and CMS software environments are simply OpenStack containers in the format of a compatible VM image. 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

One approach to generating the image is the open-source tool `packer` [8], interfaced to `puppet` [10]. `Packer` allows to configure an image based on an `.iso` file using a kickstart [9] file and flexible script-based configuration. It also provides an interface to `puppet` making it a convenient tool 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 require an initial manual start of a machine with the same hostname to allow handshake signing of the certificate from the `puppet` server.

Apart from these initial adjustments, `packer`'s interface to `puppet` allows a fully automated image generation with up-to-date configuration.

3.2 Image generation using the Oz toolkit

Another option to employ a fully-automated procedure is to use the OZ toolkit [4]. 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. Using 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 virtualized HPC systems and commercial cloud offerings provide 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. While some cloud providers, for example Amazon with AWS Batch [5], provide a service for workflow management, 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 [2]. 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 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, 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 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 HTCondor project provides a workload management system which is highly configurable and modular [6]. 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 to connect from within isolated network zones, for example via a Network Address Translation Portal, to the central HTCondor pool are available. The Connection Brokering (CCB) service [7] 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

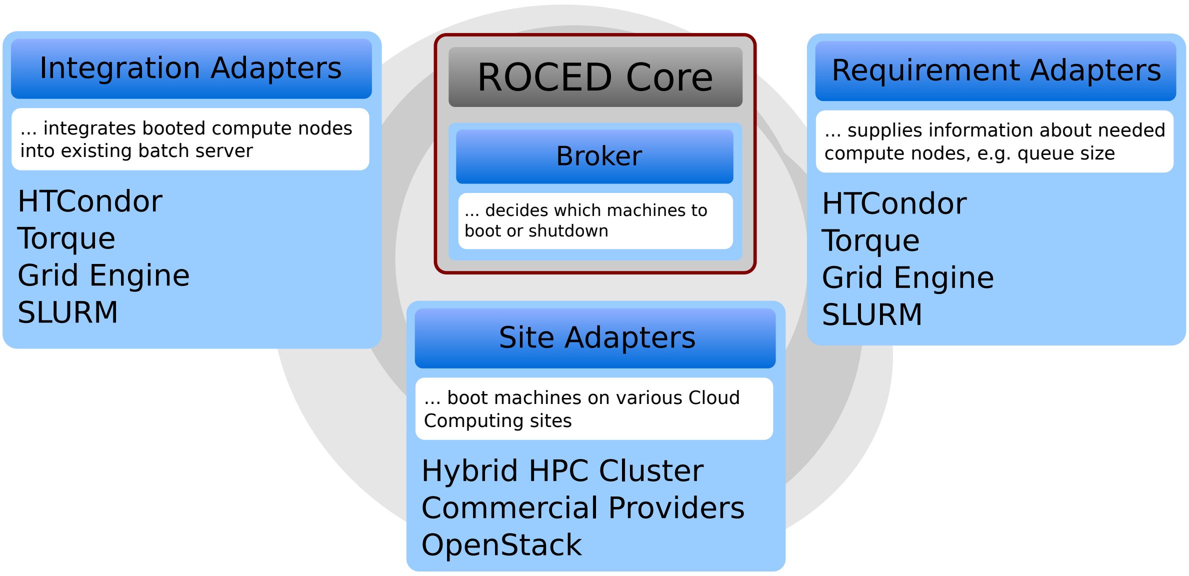

Fig. 1 Overview of the ROCED modular design. The ROCED Core contains the Broker which decides when and on which sites new virtual machines are booted. The Requirement Adapters report about the utilization and resource requirements of the attached batch systems. The Site Adapter is responsible to manage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

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 in previous seciton, the scheduler SLURM has been incorporated 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 [11], this has been 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 the 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.

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 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

The approach described above has been implemented and put into production in the research groups at the

University of Freiburg (Physikalisches Institut) 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 performance and usage statistics.

5.1 Benchmarks

Besides the use of the legacy HEP-SPEC06 (HS06) benchmark [13], the performance of the compute resources is furthermore evaluated with two fast benchmarks developed to provide real-time information of the WLCG performance and available in the CERN benchmark suite [14]; the Dirac Benchmark 2012 DB12 [15] and the ATLAS Kit Validation KV [16]. The DB12 program is evaluating the performance of CPUs through floating-point arithmetic operations, while the KV benchmark is making use of the simulation toolkit GEANT4 [17] to simulate the interactions of single muon events in the detector of the ATLAS experiment. 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. The DB12 benchmark is using the HS06 units, and the KV output provides the number of events produced per second.

To assess the impact of the virtualisation, 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 local ATLAS Tier-3 centre in Freiburg (ATLAS-BFG bare metal), or as virtual machines on the NEMO cluster (NEMO VM). On the ATLAS-BFG bare metal and on the virtual machines running on the NEMO cluster, hyper-threading (HT) technology is activated. Both are using Scientific Linux 6 [18] 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 [19]. The scores of the HEP-SPEC06, DB12, 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-BFG 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 results have been determined by step of two core units. 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.

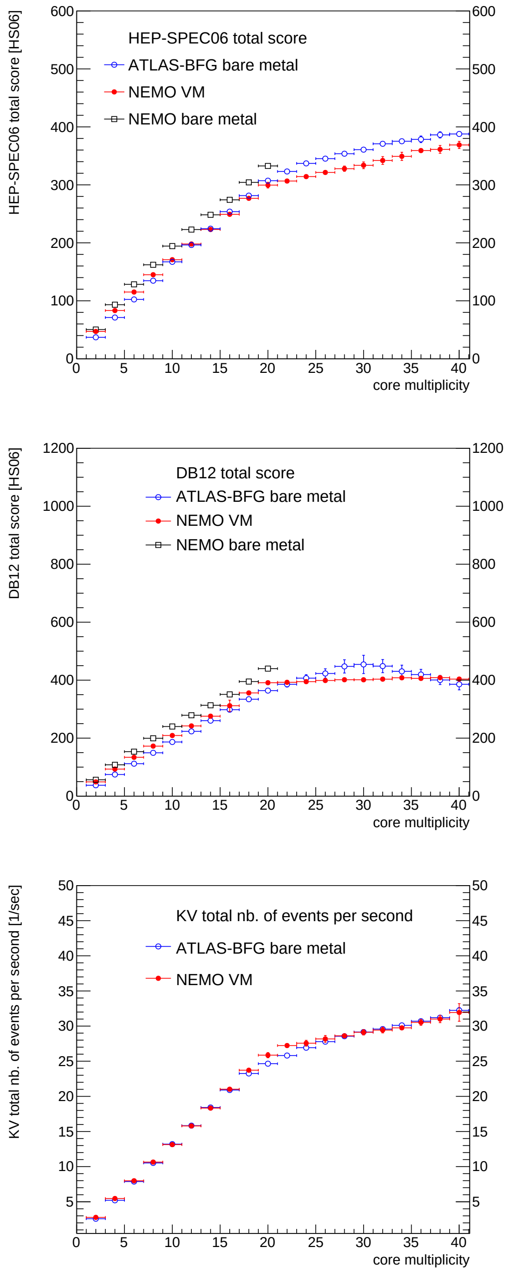

Fig. 2 Total score as a function of the core multiplicity for the HEP-SPEC06 (top), DB12 (middle) and KV (bottom) benchmarks for the ATLAS-BFG bare metal (blue open circles), the NEMO VMs (red full circles) and the NEMO bare metal (black open squares). The data points represent the average values of the benchmarks for each core multiplicity, and the vertical bars show the associated standard deviations.

The total scores are presented in figure 2 for the three benchmarks and the three configurations considered, except for the KV software for which the NEMO bare metal results are not yet available. The total scores observed for the different benchmarks are increasing until the maximum number of physical cores has been

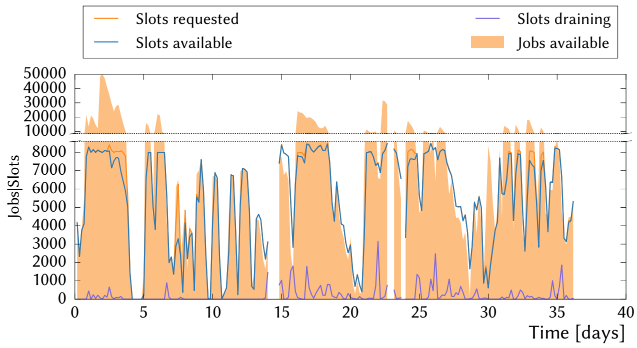

Fig. 3 Utilization of the shared HPC system by booted virtual machines. Up to 9000 virtual cores were in use at peak times. The fluctuations in the utilization reflects the patterns of the submission of jobs by our institute users. The number of draining slots displays the amount of job slots still processing jobs while the rest of the node's slot are already empty.

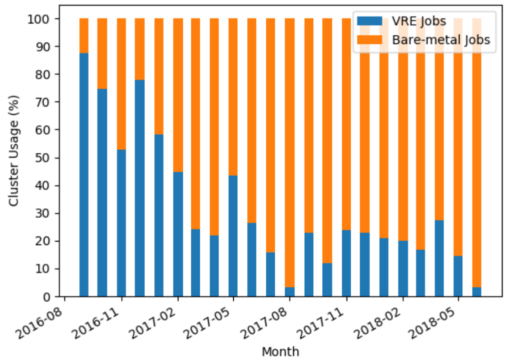

Fig. 4 Usage of the NEMO cluster in the time between September 2016 to June 2018. The blue bars indicate the usage by jobs running directly in the hosts' operating system, while the orange bars are jobs running in virtual machines.

reached, and are characterized by a flattening increase or a constant behaviour afterwards. The benchmark scores of the virtual machines running on the NEMO cluster are only slightly lower than those obtained for the NEMO bare metal operation, and the lost of performance due to the virtualisation does not exceed 10%. For the ATLAS-BFG bare metal and the VMs running on the NEMO cluster, the interplay between the virtualisation and the different operating systems leads to similar benchmark behaviours for the two configurations.

5.2 Usage statistics

Fig. 3 show 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. 4. 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 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. It 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.

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 University of Freiburg has been described. Reasons for the need for an interface between the schedulers of the host system and the external system from which requests are sent have been explained. The performance and usage have been analyzed for two groups.

This approach can be generalized to other platforms and possibly also other forms of virtualized environments (containers).

The caveat is that there is a performance loss due to the virtualization overhead. However, this is perfectly acceptable if the benefits of virtualized research environments outweigh the loss in performance for an individual use case or an entire scientific work flow. General feasibility and provisioning of virtualized research environments for scientific computing.

Using virtualization inside an HPC system opens up the possibilities for several interesting features. While their implementation would require tighter integration between HPC scheduler and virtualization framework, they could solve several classic problems with HPC systems, especially those designated for novice HPC users.

Provided the virtualization basics snapshot and migration functionality for running virtual machine instances should become further features of a virtualized cluster. This means that running processes can be stopped, possibly moved to a different node in the virtualization cluster and then 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, checkpointed 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.

References

1. OpenStack Open Source Cloud Computing Software `https://www.openstack.org/`, accessed 2018-07-03
2. ROCED Cloud Meta-Scheduler project website `https://github.com/roced-scheduler/ROCED`, accessed 2018-07-03
3. Adaptive Computing Moab `http://www.adaptivecomputing.com/moab-hpc-basic-edition/`, accessed 2018-07-03
4. Oz image generation toolkit `https://github.com/clalancette/oz`, accessed 2018-07-03
5. Amazon AWS Batch `https://aws.amazon.com/batch/`, accessed 2018-07-03
6. HTCondor workload manager `https://research.cs.wisc.edu/htcondor/`, accessed 2018-07-03
7. HTCondor Connection Brokering `http://research.cs.wisc.edu/htcondor/manual/v8.6/3_9Networking_includes.html`, 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. Slurm Elastic Computing `https://slurm.schedmd.com/elastic_computing.html`, accessed 2018-07-03
12. Dirk von Suchodoletz, Bernd Wiebelt, Konrad Meier, Michael Janczyk, Flexible HPC: bwForCluster NEMO, Proceedings of the 3rd bwHPC-Symposium: Heidelberg 2016, `http://books.ub.uni-heidelberg.de/heibooks/reader/download/308/308-4-79237-1-10-20171002.pdf`
13. HEPiX Benchmarking Working Group: `https://twiki.cern.ch/twiki/bin/view/FIOgroup/TsiBenchHEPSPEC`, accessed 2018-01-29
14. 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
15. Graciani, Ricardo and Andrew McNab, Dirac benchmark 2012, `https://gitlab.cern.ch/mcnab/dirac-benchmark/tree/master`
16. 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
17. 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
18. Fermilab and CERN, "Scientific Linux 6", `http://www.scientificlinux.org/`
19. The CentOS Project, "CentOS Linux 7", `https://www.centos.org/`
20. 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
21. T. Sjöstrand 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
22. TODO