Integration of NEMO into an existing particle physics environment through virtualization

Felix Bührer[*] Anton J. Gamel[*†] Benoît Roland[*]
Benjamin Rottler[*] Markus Schumacher[*] Ulrike Schnoor[*§]

[*]Institute of Physics, University of Freiburg, Freiburg, Germany
[†]Computing Center, University of Freiburg, Freiburg, Germany
[§]Now at CERN, Geneva, Switzerland

With the ever-growing amount of data collected with the experiments at the Large Hadron Collider (LHC) (Evans et al., 2008), the need for computing resources that can handle the analysis of this data is also rapidly increasing. This increase will even be amplified after upgrading to the High Luminosity LHC (Apollinari et al., 2017). High-Performance Computing (HPC) and other cluster computing resources provided by universities can be useful supplements to the resources dedicated to the experiment as part of the Worldwide LHC Computing Grid (WLCG) (Eck et al., 2005) for data analysis and production of simulated event samples. Computing resources in the WLCG are structured in four layers – so-called Tiers. The first layer comprises two Tier-0 computing centres located at CERN in Geneva, Switzerland and at the Wigner Research Centre for Physics in Budapest, Hungary. The second layer consists of thirteen Tier-1 centres, followed by 160 Tier-2 sites, which are typically universities and other scientific institutes. The final layer are Tier-3 sites which are directly used by local users. The University of Freiburg is operating a combined Tier-2/Tier-3, the ATLAS-BFG (Backofen et al., 2006). The shared HPC cluster »NEMO« at the University of Freiburg has been made available to local ATLAS (Aad et al., 2008) users through the provisioning of virtual machines incorporating the ATLAS software

environment analogously to the bare-metal system at the Tier-3. In addition to the provisioning of the virtual environment, the on-demand integration of these resources into the Tier-3 scheduler in a dynamic way is described. In order to provide the external NEMO resources to the user in a transparent way, an intermediate layer connecting the two batch systems is put into place. This resource scheduler monitors requirements on the user-facing system and requests resources on the backend-system.

1 Introduction

Compute clusters shared between many users at the same time cannot follow a simple first-in-first-out scheme for deciding, how resources are allocated to the work packages submitted by the users, but require sophisticated scheduling algorithms. These algorithms can take into account a wide array of parameters, both concerning the requirements of the job to be scheduled and ensuring a fair use of the provided resources.

The details of how the scheduling is done as well as the hardware being used depend on the general purpose of the cluster. In the case of targeting a High-Throughput Computing (HTC) setup, the aim is to get as much absolute compute power as possible, whereas a High-Performance Computing (HPC) setup is built in order to get results quickly using multi-node parallel-processing.

HPC, as realized in NEMO, requires a fast interconnect, a fast cluster filesystem, homogeneous machine types and no hyperthreading of the CPUs. In contrast, for HTC local storage or file caches and heterogeneous machine types may be sufficient and the network requirements are significantly reduced. Usually, hyperthreading is activated. On the operating system (OS) side, most general-purpose Linux systems can easily serve both HPC and HTC setups. The used batch schedulers on the other hand are more specifically chosen for the desired working model. An overview of many of the current cluster scheduling systems can be found in (Reuther et al., 2017). In general, HPC clusters can be used more easily for HTC-like workflows rather than vice versa. Most of the available schedulers offer possibilities to do some kind of dynamic extension or reduction of the available computing resources. These can either be only temporarily available local or remote resources.

The task at hand is to link together two independent clusters, the ATLAS-BFG and NEMO cluster, each with their own resources and two separate batch managers.

However, there are no standard interfaces or abstraction layers in place, and the cross-linking of clusters with different HPC/HTC setups, different resource managers or different login schemes is therefore not a straight-forward task. Accepting workloads from secondary schedulers or delivering basic monitoring information that can be passed through, are not features readily available today.

In order to achieve the on-demand scheduling of resources on one cluster due to requirements on a different cluster, an intermediate layer, or resource scheduler, is put in place.

From its primary concept, the NEMO HPC cluster was designed to provide a full virtualization solution with OpenStack (OpenStack Foundation, 2010) that enables users to spawn virtual machines (VMs) with a pre-configured image – so-called virtual research environments (VREs) (Suchodoletz et al., 2017). These VMs are requested by sending a wrapper-job to the NEMO batch-system (MOAB), which is queued in the same way as other jobs by NEMO users. When the wrapper-job starts, a virtual machine is spawned on the OpenStack instance. The lifetime of the VM is defined by the walltime of the MOAB job. After start-up and some initial checks, the VM is incorporated in the front-end scheduler on the ATLAS-BFG as an additional resource. This resource is in turn used to run the jobs that triggered the start of the VM in the first place. All of these mechanisms are completely transparent to the user of the ATLAS-BFG (Gamel et al., 2017).

In the following, we describe how these virtual resources are integrated into the ATLAS Tier2/Tier3 (ATLAS-BFG) cluster. NEMO and the ATLAS-BFG are utilizing different batch systems. On the user-facing (or frontend) side, SLURM (Jette et al., 2002) is used as a scheduler, while the NEMO cluster (backend) runs a combination of MOAB & Torque (Adaptive Computing, 2014). We use ROCED (Erli et al., 2017), developed at the Karlsruhe Institute of Technology (KIT) to schedule resources. ROCED is used already to integrate NEMO resources into the HTCondor (University of Wisconsin – Madison, 2018) system at KIT. Due to the modular architecture of ROCED, it can also be used for connecting the two systems in Freiburg. To do so, a new component monitoring the SLURM queue on the frontend has been developed.

The system described is running very stable and is in use by the local ATLAS researchers since fall 2017. The performance of the system has been measured using several different benchmarking programs. These benchmarks are also used to quantify the modification of the performance due to changes in the configuration of the resource scheduler and of the virtual machines being spawned. They will also be part of a future continuous monitoring effort in order to be able to detect changes in the submitted workloads. This monitoring and tuning effort will ensure a robust but also dynamic and efficient setup, that reflects changes in user workflows and requirements.

2 Challenges

The ATLAS research groups in Freiburg have very specific requirements to the operating system as well as the installed software. This is to ensure reliable scientific results across all grid sites of the WLCG.

Virtualization has been found to be a technology that can simplify the challenge to provide a specific environment on a range of different heterogeneous and changing platforms, especially in the context of particle physics (Buncic, Aguado Sánchez et al., 2011).

Being only one of multiple user groups on a shared HPC system, especially the choice of operating system has to take into account considerations from all user groups as well as from the party operating the cluster. A fully virtualized environment, independent of the choices made on the HPC cluster itself, will give the best possible scope to implement a system, that looks and behaves in the same way as the non-virtualized ATLAS-BFG cluster. This consistency between the two systems would also make it possible in the future to redirect ATLAS grid jobs submitted remotely to either NEMO or any other opportunistic resource as long as the resource provides the needed infrastructure to run the VM images. The VM images which are made available to OpenStack on NEMO have to be created and updated easily in an automatic procedure and have to fulfil the following requirements:

- Scientific Linux 6 (Fermilab et al., 2011) – current OS on the ATLAS-BFG cluster
- Access to ATLAS software via the CERN virtual file system CVMFS
- User environment from ATLAS-BFG
- Access to both grid-aware datasets on the distributed storage system dCache (Millar et al., 2014) and the local NEMO parallel filesystem BeeGFS (ThinkParQ et al., 2014)

Since the VMs are completely self-contained, all features needed to monitor and benchmark the machine are independent of the two schedulers that are involved and can either be implemented on the VM itself or offloaded to the resource scheduler. In the future, this information will also be used for continuous monitoring of the robustness and performance of the system.

Since the virtualized environment provides access to all resources in the same way as the ATLAS-BFG system, the users of the frontend system do not have to be registered as users of the backend system providing the resources.

2.1 Generation of the virtual machines

The VM template is generated with packer (HashiCorp, 2013), using a Scientific Linux 6 netinstall image as base. The customization and configuration of the template is done with puppet (Puppet, 2005), which is also used for the contextualisation of the non-virtualized worker nodes on the ATLAS-BFG cluster. Changes in configuration are automatically picked up by both systems. The output of this procedure is a static image that can be uploaded to the OpenStack server and is directly available.

In order to simplify software configurations across grid sites, most commonly-used software packages for the HEP-workflow are distributed using CVMFS (Blomer et al., 2015). The software distributed on CVMFS is managed centrally by the experiments, hosted on web servers and transferred to the worker nodes on demand.

The VMs do not contain a predefined CVMFS cache and do not utilize the hard disk as persistent cache. Instead, a RAM disk is being filled on demand from the local frontier squid (Blumenfeld et al., 2008) proxies. The RAM disk is adequately sized to cache the software used in a common ATLAS analysis. This circumvents an on-disk CVMFS installation on the host, which would lead to a heavy usage of the SSDs from mainly unused data. This is a different approach to other solutions relying on access to software from CVMFS and using an on-disk cache like CernVM (Buncic, Aguado-Sanchez et al., 2011).

2.2 Connection of front and backend batch systems

The biggest challenge for a smooth operation is the interconnection between the two different batch systems – SLURM on the frontend (ATLAS-BFG) and MOAB on the backend (NEMO) through a resource scheduler.

The workflow is as follows: the resource scheduler monitors the frontend scheduler to which users send their workload. The requirements are then compared to a list of available configurations on the NEMO HPC and the resource scheduler decides on the number of virtual machines for each configuration needed to fulfil the requirements. The appropriate number of batch jobs are sent to the backend scheduler, each spawning a virtual machine of the chosen type. The jobs that are used to start VMs in the OpenStack environment are regular user jobs on NEMO, which are started according to the availability of resources and the fair share of the NEMO user used to reproduce the fair share of the project. After startup of the VMs, they are integrated as additional resources into the SLURM scheduler and can then be used to process the user jobs queued in the frontend scheduler.

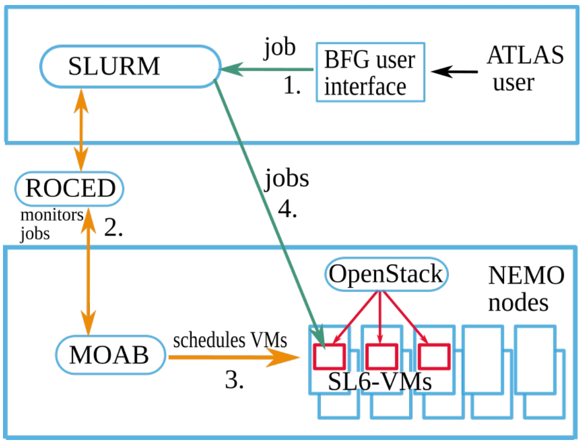

Figure 1: Schematic view of how user jobs submitted to the frontend scheduler SLURM trigger jobs to start VMs on the OpenStack instance at NEMO.

Figure 1 shows the general mode of operation. When submitting the contribution, SLURM in the ATLAS-BFG cluster is set up with separate partitions that reflect the user's affiliation to one of the ATLAS working groups. Each working group is in turn represented by a single user on NEMO, which is used to queue the jobs starting the VMs. By this mechanism, the fair shares for the different areas of research using NEMO are incorporated into the workflow.

ROCED monitors these partitions and requests the start of a VM after the user's job submission. As long as resources are requested and available on NEMO, additional virtual machines can be started. This mechanism leads to a dynamic extension of the amount of job slots available for physics analyses on the frontend system. Before VMs are integrated into SLURM, a diagnosis-check is done to see whether all needed resources are available. After a successful check the VM is set to online in SLURM and jobs can be submitted to the resources.

3 Benchmarks

To understand the performance losses introduced by going to a fully virtualized environment, different benchmarks have been run. All benchmarks are carried out on the same hardware and the results obtained on the virtualized research environment are compared to the results running directly on hardware (»bare metal«) on both the ATLAS-BFG and the NEMO cluster as well, to also assess the impact of different operating systems on the benchmark results.

In addition to the legacy HEP-SPEC06 (HS06) benchmark (HEPiX Benchmarking Working Group, 2006), the evaluation of the performance of the compute resources makes use of three benchmarking programs available in the CERN benchmark suite (Alef et al., 2017):

1. Dirac Benchmark 2012 DB12 (Graciani et al., 2012)
2. Whetstone benchmark (Curnow et al., 1976)
3. Kit Validation benchmark KV (Salvo et al., 2010).

The DB12 and Whetstone programs are evaluating the performance of the CPU through floating-point arithmetic operations. One of the main differences between the two benchmarks resides in the variables used as input to the arithmetic operations: DB12 uses random numbers generated according to a Gaussian distribution, while Whetstone utilizes variables with predefined values.

The KV benchmark runs the ATLAS software ATHENA (Calafiura et al., 2005) to simulate and reconstruct the interactions of muons in the detector of the ATLAS experiment. As our primary target is to measure performances of CPUs in the context of High Energy Physics (HEP) applications, the KV benchmark constitutes a realistic payload, more suited to our goal than the DB12 and Whetstone software. The DB12 benchmark is measured in units of HS06 and therefore can be compared directly to the results from the HEPS-SPEC06 benchmark. The Whetstone scores are expressed in Million of Whetstone Instructions Per Second (MWIPS), and the KV output provides the number of events produced per second. The different benchmarks are used to evaluate the performance of identical 20 cores Intel Xeon E5-2630 CPUs on the two different clusters: the Tier2/Tier3 cluster (ATLAS-BFG) and the shared HPC cluster (NEMO). The performance has been evaluated on three different configurations; the Tier2/Tier3 and NEMO HPC clusters running both on bare metal and the virtual machines running on the NEMO HPC cluster – except for the KV benchmark, which currently cannot be run on NEMO bare-metal nodes.

On the Tier2/Tier3, hyperthreading (HT) technology is activated and the number of cores that can be used is higher by a factor of two with respect to the physical number of CPU cores available. For the virtual machines, an arbitrary number of CPU cores can be requested. The operating system used is Scientific Linux 6 in both cases. The NEMO bare metal has no HT activated due to the more general use case of the system, and uses CentOS7 (CentOS Project, 2017) as operating system. The scores of the HEP-SPEC06, DB12, Whetstone 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 Tier2/Tier3 bare metal and for the VMs running on the NEMO cluster, 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 root-mean-squares (RMS) of the corresponding distributions have been extracted.

The scores per CPU and the total scores are presented in Figures 2 and 3 respectively, for the four benchmarks and the three configurations considered, except for the KV software for which the NEMO bare-metal results are not yet available.

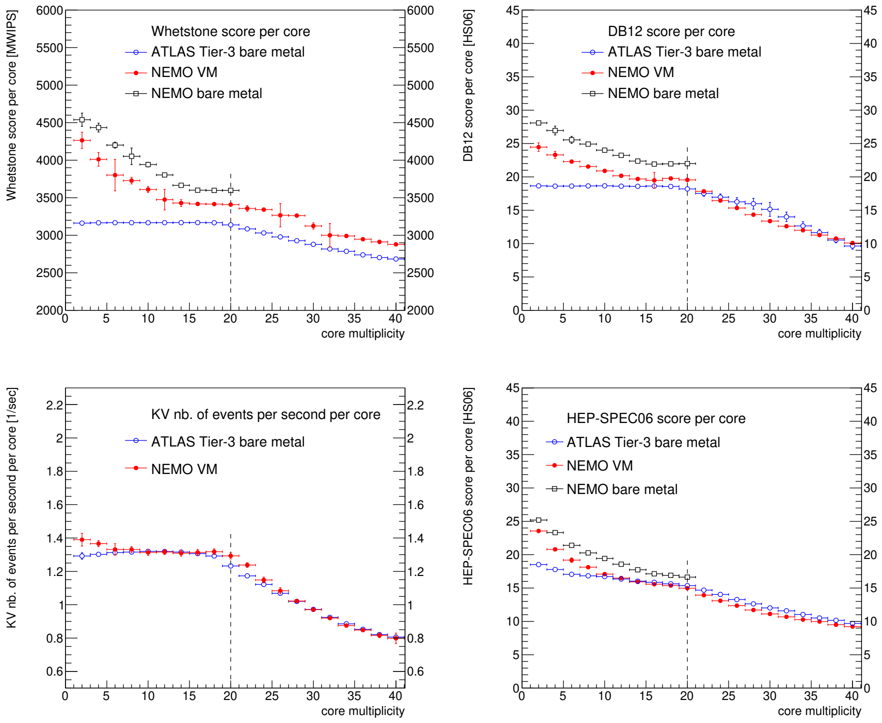

Figure 2: Score per CPU as a function of the core multiplicity for the Whetstone (top left), DB12 (top right), KV (bottom left) and HEP-SPEC06 (bottom right) benchmarks for the Tier2/Tier3 (ATLAS-BFG) running bare metal (blue open circles), the NEMO VMs (red full circles) and the NEMO bare metal (black open squares). The data points represent the average values of the benchmarks for each core multiplicity, and the vertical bars show the associated root-mean-squares. Horizontal error bars are only drawn for visibility and do not represent an uncertainty. The dotted vertical lines at a core multiplicity of 20 indicate the maximum number of physical cores.

While a constant CPU performance is expected per physical core, an increase of the score per CPU is observed in Figure 2 when going towards low values of the core multiplicity for the Whetstone, DB12 and HEP-SPEC06 benchmarks in the NEMO VMs and NEMO bare-metal configurations. Such a behaviour could be explained by a dynamic overclocking of the CPU cores in the system if not all cores are allocated. This is currently under investigation.

For the Tier2/Tier3 bare metal, the scores per CPU remain constant until the maximum number of physical cores is reached, and only start to decrease in the region where hyperthreading is active. The KV results exhibit a similar behaviour

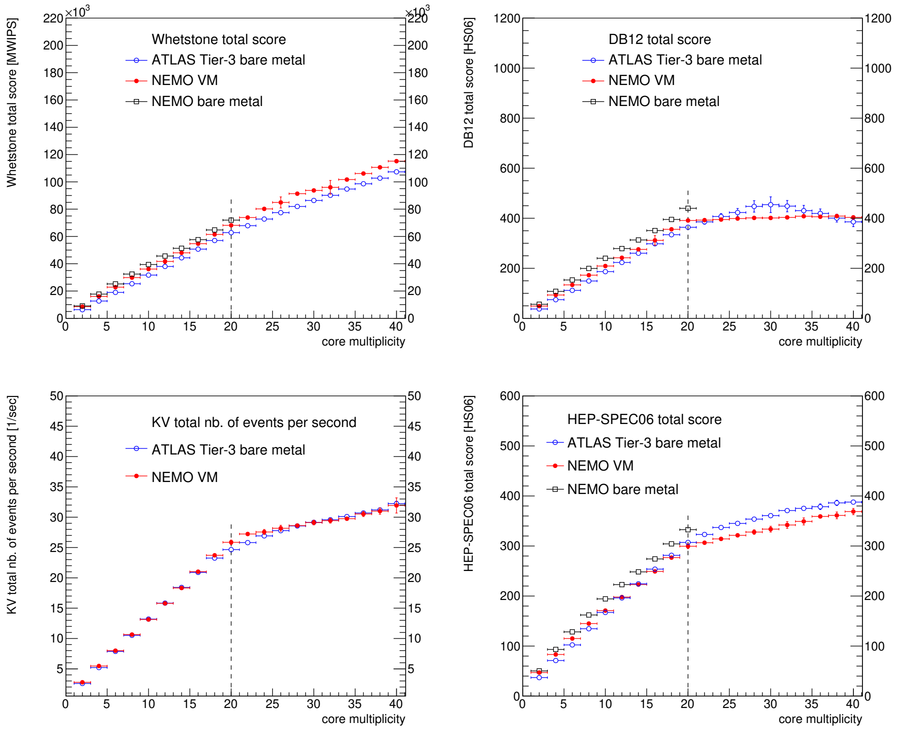

Figure 3: Total score as a function of the core multiplicity for the Whetstone (top left), DB12 (top right), KV (bottom left) and HEP-SPEC06 (bottom right) benchmarks for the Tier2/Tier3 (ATLAS-BFG) running bare metal (blue open circles), the NEMO VMs (red full circles) and the NEMO bare metal (black open squares). The data points represent the average values of the benchmarks for each core multiplicity, and the vertical bars show the associated root-mean-squares. Horizontal error bars are only drawn for visibility and do not represent an uncertainty. The dotted vertical lines at a core multiplicity of 20 indicate the maximum number of physical cores.

for both the Tier2/Tier3 bare metal and the NEMO VMs, with a constant number of events produced per second per CPU below the maximum number of physical cores and a decrease of the performance afterwards. The Whetstone score per CPU at a core multiplicity of 20, the maximum number of physical cores available, is considered as an illustrative example of the benchmark behaviours on the three different configurations. An increase of the CPU performance by the order of 5% is observed when going from the Tier2/Tier3 bare metal to the NEMO VMs, while going from the NEMO VMs to the NEMO bare metal leads to a further increase of performance of the order of 5% as well.

A continuously increasing total score is observed in Figure 3 for the Whetstone benchmark on the three different configurations, while the DB12, KV and HEP-SPEC06 results are characterized by a flattening increase or a constant behaviour once the maximum number of physical cores has been reached. The Whetstone benchmark provides higher CPU performances in the HT region in comparison to the scores obtained with the three other benchmarks. The scores obtained with the KV and HEP-SPEC06 benchmarks indicate an increase of the CPU performance by 15 to 20% when going from the maximum number of physical cores to the upper edge of the HT region, while the Whetstone scores exhibit a larger increase of the order of 60%. The Tier2/Tier3 bare metal and the VMs running on the NEMO cluster share the same configuration in terms of hardware, operating system and hyperthreading. A given benchmark should therefore exhibit a similar behaviour for both configurations and show the effect caused by the virtualization as an offset. The KV benchmark, besides being the more realistic estimator of the CPU performance in the context of HEP applications, is the only benchmark for which this expectation is observed. The behaviours of the different benchmarks still need to be studied in more detail, in order to fully understand the impact of the operating system, hyperthreading and virtualization on the CPU performances.

4 Summary

The HPC cluster NEMO has successfully been integrated into the workflow of local users in Freiburg running ATLAS data analysis jobs. This has been achieved in a transparent way using full virtualization on NEMO and utilizing the resource scheduler ROCED for the integration of the virtual resources into the frontend scheduler. The system is in production since fall 2017.

First performance tests using different benchmarks show some degrading in performance of the virtual machines compared to running on bare metal. The differences are within the expected ranges when using different operating systems and are significantly reduced when going from pure CPU benchmarks like Whetstone or DB12 to benchmarks more closely related to high energy particle physics analysis like HS06 or KV.

The continuous benchmarking effort will ensure a stable and efficient environment. Integrating the results of the benchmarks into the resource-scheduling process can enable cluster administrators to test different configurations, leading to a more efficient usage of the provided resources.

Acknowledgements

The research is supported by the Ministerium für Wissenschaft, Forschung und Kunst Baden-Württemberg in the project MWK 34-7547.221 »ViCE – Virtual Open Science Collaboration Environment« and by the Bundesministerium für Bildung und Forschung in the project 05H15VFCA1 »Higgs-Physik und Grid-Computing für das ATLAS-Experiment am LHC«.

Corresponding Authors

Felix Bührer: `felix.buehrer@physik.uni-freiburg.de`
Anton J. Gamel: `anton.gamel@physik.uni-freiburg.de`
Institute of Physics, University of Freiburg, Freiburg, Germany

ORCID

Felix Bührer https://orcid.org/0000-0002-9274-5004
Anton J. Gamel https://orcid.org/0000-0002-7044-8324
Benoît Roland https://orcid.org/0000-0003-3397-6475
Benjamin Rottler https://orcid.org/0000-0002-6762-2213
Markus Schumacher https://orcid.org/0000-0002-1733-8388
Ulrike Schnoor https://orcid.org/0000-0002-2237-384X

References

Aad, G. et al. (2008). »The ATLAS Experiment at the CERN Large Hadron Collider«. In: *JINST* 3, S08003. DOI: 10.1088/1748-0221/3/08/S08003.

Adaptive Computing (2014). *MOAB HPC SUITE*. URL: http://www.adaptivecomputing.com/products/hpc-products/moab-hpc-suite-grid-option/.

Alef, M. et al. (2017). »Benchmarking cloud resources for HEP«. In: *J. Phys. Conf. Ser.* 898.9, p. 092056. DOI: 10.1088/1742-6596/898/9/092056.

Apollinari, G., O. Bruening, T. Nakamoto and L. Rossi (2017). »High Luminosity Large Hadron Collider HL-LHC«. In: *CERN Yellow Report CERN-2015-005*, pp. 1–19. DOI: `10.5170/CERN-2015-005.1`. arXiv: `1705.08830`.

Backofen, R. et al. (2006). »A Bottom-up approach to Grid-Computing at a University: the Black-Forest-Grid Initiative«. In: *Praxis der Informationsverarbeitung und Kommunikation* 29, pp. 81–87. DOI: `10.1515/PIKO.2006.81`.

Blomer, J. et al. (2015). »The Evolution of Global Scale Filesystems for Scientific Software Distribution«. In: *Computing in Science and Engineering* 17.6, pp. 61–71.

Blumenfeld, B., D. Dykstra, L. Lueking and E. Wicklund (2008). »CMS conditions data access using FroNTier«. In: *Journal of Physics: Conference Series* 119.7, p. 072007. URL: `http://stacks.iop.org/1742-6596/119/i=7/a=072007`.

Buncic, P., C. Aguado Sánchez, J. Blomer, A. Harutyunyan and M. Mudrinic (2011). »A practical approach to virtualization in HEP«. In: *The European Physical Journal Plus* 126.1, p. 13. ISSN: 2190-5444. DOI: `10.1140/epjp/i2011-11013-1`.

Buncic, P., C. Aguado-Sanchez, J. Blomer and A. Harutyunyan (2011). »CernVM: Minimal maintenance approach to virtualization«. In: *Journal of Physics: Conference Series* 331.5, p. 052004. URL: `http://stacks.iop.org/1742-6596/331/i=5/a=052004`.

Calafiura, P., W. Lavrijsen, C. Leggett, M. Marino and D. Quarrie (2005). »The Athena Control Framework in Production, New Developments and Lessons Learned«. In: *Computing in High Energy Physics and Nuclear Physics 2004*. DOI: `10.5170/CERN-2005-002.456`.

CentOS Project (2017). *CentOS Linux release 7.4.1708 (Core)*. URL: `https://www.centos.org/`.

Curnow, H. and B. Wichman (1976). »A Synthetic Benchmark«. In: *Computer Journal* 19, pp. 43–49.

Eck, C. et al. (2005). *LHC computing Grid: Technical Design Report. Version 1.06 (20 Jun 2005)*. Technical Design Report LCG. Geneva: CERN. URL: `https://cds.cern.ch/record/840543`.

Erli, G. et al. (2017). »On-demand provisioning of HEP compute resources on cloud sites and shared HPC centers«. In: *Journal of Physics: Conference Series* 898.5, p. 052021. URL: `http://stacks.iop.org/1742-6596/898/i=5/a=052021`.

Evans, L. and P. Bryant (2008). »LHC Machine«. In: *JINST* 3, S08001. DOI: `10.1088/1748-0221/3/08/S08001`.

Fermilab and CERN (2011). *Scientific Linux release 6.8 (Carbon)*. URL: `http://www.scientificlinux.org/`.

Gamel, A. J., U. Schnoor, K. Meier, F. Bührer and M. Schumacher (2017). *Virtualization of the ATLAS software environment on a shared HPC system*. Tech. rep. ATL-SOFT-PROC-2017-070. Geneva: CERN. URL: `https://cds.cern.ch/record/2292920`.

Graciani, R. and A. McNab (2012). *Dirac benchmark 2012*. URL: `https://gitlab.cern.ch/mcnab/dirac-benchmark/tree/master`.

HashiCorp (2013). *packer*. URL: `https://www.packer.io/`.

HEPiX Benchmarking Working Group (2006). *HEP SPEC06 (HS06) benchmark*. URL: `https://w3.hepix.org/benchmarking.html`.

Jette, M. A., A. B. Yoo and M. Grondona (2002). »SLURM: Simple Linux Utility for Resource Management«. In: *In Lecture Notes in Computer Science: Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP) 2003*. Springer-Verlag, pp. 44–60.

Millar, A. P. et al. (2014). »dCache: Big Data storage for HEP communities and beyond«. In: *Journal of Physics: Conference Series* 513.4, p. 042033. URL: `http://stacks.iop.org/1742-6596/513/i=4/a=042033`.

OpenStack Foundation (2010). *OpenStack*. URL: `https://www.openstack.org/`.

Puppet (2005). *puppet*. URL: `https://puppet.com`.

Reuther, A. et al. (2017). »Scalable System Scheduling for HPC and Big Data«. In: *Journal of Parallel and Distributed Computing*. DOI: `10.1016/j.jpdc.2017.06.009`. arXiv: `1705.03102`.

Salvo, A. D. and F. Brasolin (2010). »Benchmarking the ATLAS software through the Kit Validation engine«. In: *Journal of Physics: Conference Series* 219.4, p. 042037. URL: `http://stacks.iop.org/1742-6596/219/i=4/a=042037`.

Suchodoletz, D. von, B. Wiebelt, K. Meier and M. Janczyk (2017). »Flexible HPC: bwFor-Cluster NEMO«. In: *Proceedings of the 3rd bwHPC-Symposium*. (2016). Heidelberg: heiBOOKS. DOI: `10.11588/heibooks.308.418`.

ThinkParQ and ITWM (2014). *BeeGFS*. URL: `https://www.beegfs.io`.

University of Wisconsin – Madison (2018). *HTCondor*. URL: `https://research.cs.wisc.edu/htcondor/`.