EPJ Web of Conferences **214**, 07025 (2019) https://doi.org/10.1051/epjconf/201921407025
CHEP 2018

Integrating HPC into an agile and cloud-focused environment at CERN

Pablo Llopis[1,*], *Carolina* Lindqvist[1,**], *Nils* Høimyr[1,***], *Dan* van der Ster[1,****], and *Philippe* Ganz[1,†]

[1]CERN, Switzerland

Abstract. CERN's batch and grid services are mainly focused on High Throughput computing (HTC) for processing data from the Large Hadron Collider (LHC) and other experiments. However, part of the user community requires High Performance Computing (HPC) for massively parallel applications across many cores on MPI-enabled infrastructure. This contribution addresses the implementation of HPC infrastructure at CERN for Lattice QCD application development, as well as for different types of simulations for the accelerator and technology sector at CERN. Our approach has been to integrate the HPC facilities as far as possible with the HTC services in our data centre, and to take advantage of an agile infrastructure for updates, configuration and deployment. The HPC cluster has been orchestrated with the OpenStack Ironic component, and is hence managed with the same tools as the CERN internal OpenStack cloud. Experience and benchmarks of MPI applications across Infiniband with shared storage on CephFS is discussed, as well the setup of the SLURM scheduler for HPC jobs with a provision for backfill of HTC workloads.

1 Introduction

The IT Department at the European Organization for Nuclear Research (CERN) in Geneva operates a data center acting as a Tier-0 for the Worldwide LHC Grid (WLCG) to handle the massive data requirements for processing of the LHC data. Furthermore it also hosts a variety of computing services for the infrastructure of the laboratory, as well as for the user community of scientists and engineers working on current and future accelerators and experiments. While the bulk of the batch processing requirements for particle physics is handled by embarrassingly parallel applications running in a High Throughput Computing (HTC) environment, some applications require High Performance Computing (HPC) facilities for the execution of applications using MPI (Message Passing Interface), long-running jobs, and other applications that require an exceptionally large amount of memory or computing power. Notably Lattice-QCD simulations used by theorists, accelerator physics simulations and engineering applications, e.g. computational fluid dynamics and field calculations can benefit from parallel computing on clusters with low-latency interconnects.

*e-mail: pablo.llopis@cern.ch
**e-mail: carolina.linqvist@cern.ch
***e-mail: nils.hoimyr@cern.ch
****e-mail: daniel.vanderster@cern.ch
†e-mail: philippe.ganz@cern.ch

In this paper, we describe the challenges applying Agile Methods and Cloud technologies in an HPC environment, and how to balance these with a stable environment for long-running HPC simulations. The HTC batch farm at CERN currently has about 175 000 hyperthreaded cores, while the HPC clusters total about 5600 physical cores (11200 with hyperthreading). HTCondor[1] is used for CERN's HTC batch infrastructure and replaces the LSF batch system. We have chosen to deploy SLURM[2] for HPC job scheduling, due to its superior support for MPI applications in a multi-host environment. Backfill of idle HPC worker nodes is done via a HTCondor-SLURM gateway, and a dedicated Compute Element (CE) that can be used to send Grid jobs from the WLCG to our HPC resources.

2 Challenges of HPC in an Agile environment

The Agile Infrastructure[3] for computing at CERN ensures that computers in the CERN data centre are installed, configured, monitored and maintained using a stack of tools that is shared and maintained by teams across the IT department. All IT services are using this layered approach, and also earlier, smaller HPC facilities at CERN have relied on available components in the CERN IT service stack[4]. In this approach the configuration of servers is done with Puppet[5] that is by default set to run with a one hour interval to apply configuration updates. The servers are contained in hostgroups which correspond to a certain configuration in Puppet. Parts of the configuration that is common for a larger set of servers is usually contained in a Puppet module. A module can cover the configuration of a specific software or a whole software environment that is common for a groups of servers.

The operating system packages are updated daily by a cronjob that triggers updates using the standard Red Hat/CentOS yum package manager. The installation of machines is done using in-house software tools for maintaining the lifecycle of a machine, called `ai-tools`. Collectd and Grafana are used for monitoring and plotting. Each service is maintained by a dedicated team. For example storage, databases and monitoring are all handled by separate teams. Thanks to this environment, CERN-IT is able to quickly roll out security updates and configuration changes across thousands of servers in our local and remote data centre. This kind of configuration largely applies to the CERN HPC resources, with some modifications that assure the stability of the cluster. This cluster configuration is to a large extent shared with the large HTC batch farm, in order to benefit from security updates as well as the shared environment that is required to run a large variety of physics software.

2.1 Applying the agile approach to HPC

The average lifecycle of a node is also different if one compares the HTC and the HPC resources. For HTC all cluster resources are virtualized and nodes can be killed and recreated rapidly through OpenStack APIs. For HPC it is more time-consuming to recreate (reinstall) a node, and the aim is more often to keep the cluster stable and running rather than applying all available upgrades. As the bare-metal resources are heterogeneous in a visible manner, as opposed to homogeneous virtualized resources, this also needs to be considered when making changes to the software. HPC workloads and MPI software are sensitive to changes that affect the MPI software stack, notably the kernel version and the drivers for Infiniband and low-latency Ethernet. The configuration file for SLURM, especially the list of nodes that the cluster consists of, must also be identical across the cluster for the scheduler to work properly. In addition, the general server configuration, e.g. versions of packages that the simulations running on the cluster use, must be consistent to prevent program failures, as programs run on multiple nodes that communicate with each other.

In order to apply the agile approach to HPC we aim for automating as much as possible and applying a quality assurance (QA) process to changes that are made. We also aim to use RPM-packaged software and package applications ourselves if necessary. All configuration changes are set through Puppet and also go through a QA process before these changes are merged from development into QA and finally production. This allows us to control the state of the cluster and also ensures that a node can be removed and reinstalled fast, compared to a manual installation and manual changes to configuration files. It also ensures that the configuration is consistent.

On the HTC batch servers the regular QA approach is implemented by having a certain percentage of HTC batch servers running a recently updated configuration in production to allow for problem detection. This approach does not work well in the HPC clusters, as the updates need to be handled in a more controlled manner. Thus, we have addressed this issue by setting a fixed software repository date that is updated on regular intervals. A small amount of the HPC cluster nodes are allocated for QA and do not have a fixed repository date, which means they will be upgraded as soon as new system packages are released. However, these QA nodes are not visible to the user and are exclusively used for QA purposes. A green light for an update of the repository date in production, and hence the system packages, is given once the package versions have been validated on a small set of HPC QA nodes with MPI workloads. Configuration changes, i.e. changes in the Puppet-managed configuration of the nodes, also follow the same procedure of first being applied on a set of QA nodes, tested with user applications and then applied to the production environment. We plan to automate this procedure to a large extent as part of future work which is detailed in Section 5.

3 Challenges of HPC in a Cloud-focused environment

At CERN, most of the computing resources run under virtual machines in our private OpenStack cloud[6] that consists of more than 300 000 cores (over 9000 physical hosts) [7]. The resources are managed as described in Section 2. Our HTC cloud fits into this model of resource management, however the HPC cluster has some more special needs. HPC clusters and supercomputers are often custom-built with components selected for the cluster keeping high-performance in mind. The machines run a bare minimum of processes that are necessary for the cluster to function and any excessive monitoring or resource-consuming daemons are turned off. The aim is to improve the performance and efficiency of the machines. Any additional layers that run on top of the hardware is generally avoided, unless necessary. However, our HPC strategy needs to be a cost-effective solution based on existing data center and IT-infrastructure.

The way we integrate with the cloud resources is on one hand by using virtual machines for all cluster management nodes, i.e. submission-, head-, and database nodes. On the other hand, we also provision one full cluster of worker nodes and part of an older cluster using OpenStack Ironic [8]. This setup is described in detail in Section 3.1. This means that we have three different types of resources – bare-metal nodes with and without Ironic, as well as virtual machines. Many of our tools support all three types of resources, and all nodes that run under OpenStack can additionally take advantage of several OpenStack tools and APIs.

3.1 Cloud for High Performance Computation

While use of OpenStack to orchestrate virtual machines is a well established practice, the use of the OpenStack Ironic component for provisioning of bare metal servers operated under the OpenStack framework is fairly novel. Ironic has been used to bootstrap and orchestrate the

worker nodes of two HPC clusters, as well as other batch compute nodes on physical hardware. The infrastructure nodes for the SLURM batch system, as well as the submit nodes where users log in and submit their jobs, are virtual machines that also run under OpenStack; hence a uniform framework is used to orchestrate all of the HPC infrastructure. Administrative operations such as server restart, power-cycle and console operations is done via the OpenStack interfaces (Web-console and API) for both Ironic nodes and virtual machines.

3.2 Challenges in Cloud related to High Performance File I/O

At CERN, the interactive login service *lxplus* as well as the HTC batch nodes utilize AFS as a shared file system and EOS for physics data storage[9]. For the HPC use case, these storage solutions and their associated access control layers pose too much overhead, and the filesystems are not intended for high-performance parallel I/O. Block and Object-storage based on Ceph[10] is widely used for our cloud infrastructure, and CephFS shared filesystems are provided as an on-demand service through OpenStack Manila. Therefore, instead of adding support for a dedicated HPC file system like e.g. GPFS or Lustre, we have deployed CephFS as our main shared file system for the HPC service. CephFS is used as a mounted shared file system for user home directories on the HPC cluster's worker and submit nodes. Both the shared batch HPC clusters and the dedicated Theory Lattice QCD cluster utilize a CephFS home directory managed by OpenStack's Manila service.

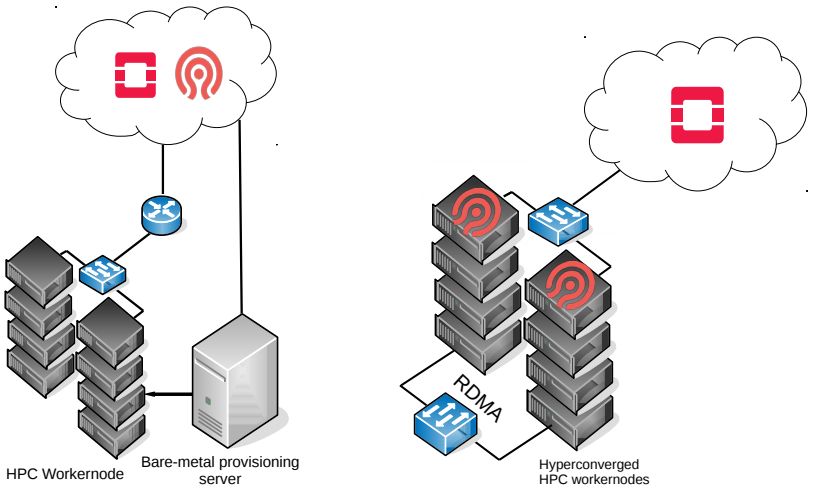
Additionally, local SSD disks on our 2 HPC Infiniband clusters have been used to build a hyperconverged CephFS cluster on the worker nodes, and hence on the same network switch. This CephFS cluster is used to provide a cluster-wide scratch space for higher parallel I/O performance. Figure 1 depicts the different strategies described in this Section. A detailed performance study of this "hyperconverged" computing and storage setup is currently underway and will be published in a future article.

4 Cluster and batch scheduling

We began setting up SLURM as the main HPC scheduler in 2016. We evaluated the Parallel universe in HTCondor as an alternative for HPC, but concluded that SLURM was more feature-rich and better suited for our purposes. It can be noted that one of the few manual processes we keep in place is adding nodes to the cluster. The reason for not automating such a seemingly simple procedure is, in short, that SLURM needs the configuration file to have the same content on all nodes, especially with regards to what nodes are part of the cluster. If such changes are not propagated within a short time, the SLURM configuration for each node may become inconsistent, resulting in SLURM malfunctioning. As the size of our resources rarely changes, we therefore rely on manual and planned interventions to carry out addition or removal of nodes for any of our clusters. While this process may involve a very short downtime of the SLURM management nodes, running jobs are unaffected by these changes.

4.1 SLURM setup

The SLURM cluster consists of four types of nodes as seen in Figure 2 – head nodes, database nodes, submit nodes, and worker nodes. All nodes except the worker nodes are VMs. The head nodes are the main management nodes of the cluster and run the slurmctld daemon that manages the SLURM cluster. The database nodes run the slurmdbd daemon which manages accounting operations and communicates with an external MySQL database for storing the data. We have set up two head nodes and two database nodes, and have the SLURM failover



(a) Classic worker nodes

(b) Hyperconverged worker and storage nodes, running compute workloads alongside CephFS services.

Figure 1: Comparison of classic (a) and hyperconverged (b) worker nodes.

feature enabled. This has proven to be a good enough setup for our scale of deployment. The submit nodes are configured as SLURM worker nodes but cannot run jobs, only submit them. The worker nodes are physical, bare-metal provisioned hosts that execute the submitted jobs. We have configured SLURM to consider our resources as two separate partitions that mainly differ with regards to storage and network interconnects. One partition contains nodes with 10Gb Ethernet whereas the other partition contains nodes that are equipped with Infiniband and hyperconverged storage as detailed in Section 3.2. The resource accounting is integrated with the general accounting group setup that is used across the HTCondor batch system at CERN. We have used the SLURM Multifactor Priority plugin [11] to set up fairshare of the resources, mainly to set quotas for how much resources each group and user account can use. All configuration is done by an open source Puppet module for SLURM[12].

4.2 Cluster software management

The operating system software is based on the standard CERN CentOS 7 image that is used by default for nodes in the CERN IT infrastructure. On top of that we add necessary software such as OFED[13] drivers for the network hardware and various implementations and versions of MPI (mpich[14], mvapich[15] and OpenMPI[16]). The reason we support many MPI implementations is that some user applications require specific versions in order to run. We use `yum` as package manager and Puppet for configuring and selecting which additional packages need to be installed. If necessary, we also package software as RPMs and build them using the Koji build system[17]. User applications are stored in project folders or personal scratch space on a shared filesystem, either on AFS or Ceph. These shared filesystems are mounted across the cluster.

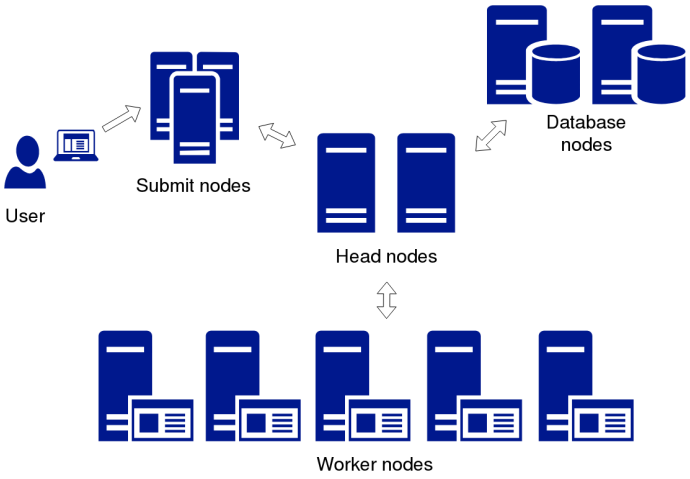


Figure 2: The SLURM cluster setup.

4.3 HPC cluster backfill

Achieving high utilization is a desired goal for HPC facilities in order to achieve an efficient use of computing resources. However, keeping a high resource utilization over a prolonged period of time is challenging due to various reasons. First, user demand varies and there may be both peaks and lows, causing periods of time where there are not enough user job submissions to fully utilize all resources. Second, even if there are sufficient submissions to create a long waiting queue to run, there may still be idle nodes. This may happen when the requirements of queued jobs do not match the available nodes. Frequently, there are idle resources due to jobs requiring more nodes than those available. Therefore, as the SLURM HPC cluster is not always fully utilized, our goal is to enable the backfilling of HTCondor jobs into idle SLURM resource. A backfill gateway that combines the HTCondor Computing Element with both the Condor GridManager and the Blahp gateway to launch backfill jobs on the SLURM cluster is currently under testing.

4.4 Containerizing HPC applications

As some users of the cluster may require a more tailored version of the environment where their application runs, we investigated suitable container technologies and began supporting Singularity containers [18]. The well-known container platform Docker was ruled out as it requires root privileges to run and involves more work when setting up low-latency networking for inter-node MPI communications. Among the benefits of Singularity, we highlight that it integrates well with SLURM and also supports Docker images. Our users can launch Singularity containers through SLURM and we already provide images for certain user applications that require a special setup or libraries to run. Some engineering applications, notably Fluent[19], provide their software packaged as a Docker container, which also can run on our cluster. However, the containers need to be set up and maintained, and even if a software is already released in a container it may not work out of the box. Another drawback of containers is the fact that the responsibility of what is running inside the container is pushed to the user. While this may seem as an advantage to the system administrator, who sees this burden shifted towards the user, some users may not be experienced enough to handle this task so it

may backfire as additional work to be performed by the system administrators to develop and maintain container images. However, the advantage of isolation remains when compared to classic non-containerized jobs, as it is easier to create independent, non-interfering environments with different software stacks. In our experience, the more software dependencies an application requires, the more beneficial the use of containers may become.

5 Future Work

The main focus of the future work for the HPC service is the improvement of cluster performance and increasing automation. As part of ongoing user support we may suggest improvements for how user applications are launched, e.g. by providing containers or suggesting changes to the parameters that the users have set for their job. We consider two approaches to increasing the efficiency and the reliability of the HPC service, as detailed in the following subsections.

5.1 Automating gathering of performance-related metrics

We lack a systematic way of collecting and analyzing performance metrics of cluster workloads. These metrics and their respective analysis would enable making data-driven decisions to improve resource utilization. Resource utilization would be increased on one hand by improving job efficiency through submission parameters (such as number of task, ratio of tasks per node, and task placement); on the other hand through system tuning using insights obtained from detailed workload and system analysis.

5.2 Automated test framework

Another tool that is planned to be implemented is an automated test framework that would run real HPC workloads on idle resources for testing purposes. To improve the stability of the cluster and ensure that no unverified changes are applied, we plan to create an automated test framework to run real HPC workloads periodically in the cluster. This would allow us to see, e.g., performance changes that appear in the cluster after an upgrade, changes that break user applications, and in general work as part of the service monitoring that ensures that our cluster is up and running. For the batch cluster a solution called HammerCloud[20] is already in place. In the ideal case we would be able to test all installed MPI versions and flavors, as well as user applications that are bound to specific MPI versions. The test results would be stored externally to the cluster, and presented on a monitoring dashboard with alerts for any tests that break.

Acknowledgements

We would like to thank the anonymous reviewers for improving the quality of this submission.

References

[1] D. Thain, T. Tannenbaum, M. Livny, Concurrency and computation: practice and experience **17**, 323 (2005)

[2] A.B. Yoo, M.A. Jette, M. Grondona, *Slurm: Simple linux utility for resource management*, in *Workshop on Job Scheduling Strategies for Parallel Processing* (Springer, 2003), pp. 44–60

[3] B. Jones, G. McCance, S. Traylen, N.B. Arias, **664**, 022026 (2015)
[4] M. Husejko, N. Høimyr, A. Gonzalez, G. Koloventzos, D. Asbury, A. Trzcinska, I. Agtzidis, G. Botrel, J. Otto, **513**, 052012 (2014)
[5] T. Uphill, *Mastering Puppet* (Packt Publishing Ltd, 2016)
[6] T. Bell, B. Bompastor, S. Bukowiec, J.C. Leon, M. Denis, J. van Eldik, M.F. Lobo, L.F. Alvarez, D.F. Rodriguez, A. Marino et al., **664**, 022003 (2015)
[7] A. Wiebalk (2018), (R)Evolution in CERN IT Operational War Stories from 5 Years of Running OpenStack in Production, Openstack Summit Vancouver 2018, https://www.youtube.com/watch?v=3HjQmWYp1Sk
[8] OpenStack Technical Committee, *Ironic - OpenStack*, https://wiki.openstack.org/wiki/Ironic (2018), accessed: 2018-10-23
[9] X. Espinal, E. Bocchi, B. Chan, A. Fiorot, J. Iven, G.L. Presti, J. Lopez, H. Gonzalez, M. Lamanna, L. Mascetti et al., **898**, 062028 (2017)
[10] S.A. Weil, S.A. Brandt, E.L. Miller, D.D. Long, C. Maltzahn, *Ceph: A scalable, high-performance distributed file system*, in *Proceedings of the 7th symposium on Operating systems design and implementation* (USENIX Association, 2006), pp. 307–320
[11] SLURM, *Multifactor Priority Plugin*, https://slurm.schedmd.com/priority_multifactor.html (2018), accessed: 2018-10-30
[12] P. Ganz, C. Lindqvist, P. Llopis, *Puppet-slurm*, https://github.com/cernops/puppet-slurm (2018), accessed: 2018-10-30
[13] OpenFabrics Enterprise Distribution, *OFED*, https://downloads.openfabrics.org/OFED/ (2018), accessed: 2018-10-30
[14] MPICH, *MPICH overview*, https://www.mpich.org/about/overview/ (2018), accessed: 2018-10-30
[15] MVAPICH, *MVAPICH :: Overview*, http://mvapich.cse.ohio-state.edu/overview (2018), accessed: 2018-10-30
[16] OpenMPI, *General information about the OpenMPI project*, https://www.open-mpi.org/faq/?category=general (2018), accessed: 2018-10-30
[17] McLean, Mike et al., *Overview - Koji*, https://pagure.io/koji (2018), accessed: 2018-10-30
[18] A. Wennersteen, *Evaluation of Containers for HPC* (2018), https://doi.org/10.5281/zenodo.1438401
[19] Fluent Project, *Fluent Docker Image*, https://hub.docker.com/r/fluent/fluentd/ (2018), accessed: 2018-10-30
[20] J. Schovancova, A. Di Girolamo, A. Fkiaras, V. Mancinelli, Tech. rep. (2018)