On-demand provisioning of HEP compute resources on cloud sites and shared HPC centers

On-demand provisioning of HEP compute resources on cloud sites and shared HPC centers

G Erli[1], F Fischer[1], G Fleig[1], M Giffels[1], T Hauth[1], G Quast[1], M Schnepf[1], J Heese[2], K Leppert[2], J Arnaez de Pedro[2], R Sträter[2]

[1] Karlsruhe Institute of Technology, Institute of Experimental Nuclear Physics (IEKP), Wolfgang-Gaede-Str. 1, 76131 Karlsruhe, Germany

[2] 1&1 Internet SE, Elgendorfer Str. 57, 56410 Montabaur, Germany

E-mail: `thomas.hauth@kit.edu, joerg.heese@1und1.de`

Abstract. This contribution reports on solutions, experiences and recent developments with the dynamic, on-demand provisioning of remote computing resources for analysis and simulation workflows. Local resources of a physics institute are extended by private and commercial cloud sites, ranging from the inclusion of desktop clusters over institute clusters to HPC centers.

Rather than relying on dedicated HEP computing centers, it is nowadays more reasonable and flexible to utilize remote computing capacity via virtualization techniques or container concepts.

We report on recent experience from incorporating a remote HPC center (NEMO Cluster, Freiburg University) and resources dynamically requested from the commercial provider 1&1 Internet SE into our intitute's computing infrastructure.

The Freiburg HPC resources are requested via the standard batch system, allowing HPC and HEP applications to be executed simultaneously, such that regular batch jobs run side by side to virtual machines managed via OpenStack [1]. For the inclusion of the 1&1 commercial resources, a Python API and SDK as well as the possibility to upload images were available. Large scale tests prove the capability to serve the scientific use case in the European 1&1 datacenters.

The described environment at the Institute of Experimental Nuclear Physics (IEKP) at KIT serves the needs of researchers participating in the CMS and Belle II experiments. In total, resources exceeding half a million CPU hours have been provided by remote sites.

1. Introduction

Grid computing in general and the Worldwide LHC Computing Grid (WLCG) specifically are an extremely successful implementation of the concept of distributed computing and storage and have enabled spectacular discoveries in recent years. The most prominent is arguably the confirmation of the Higgs Boson by the CMS and ATLAS Collaborations at the LHC in the year 2012.

One thing these experiments have in common is that enormous amounts of computation and storage requirements need to be fulfilled to achieve the best possible scientific output. Data recorded by these experiments needs to be reconstructed, stored and analyzed. Furthermore, Monte-Carlo simulations need to be performed in order to compare the recorded data to model predictions. Depending on the experiment, at least a comparable amount of simulated events needs to be available in order to achieve a sufficient statistical accuracy. In practice, many experiments need two or three times more simulated events than recorded ones in order to test for different models. With the increased data rates of the next-generation HEP experiments, the computing requirements will also grow in the coming years.

Classically, the Grid infrastructure is installed on dedicated hardware resources which are located in data centers and tailored to serve as Grid computing sites. This allows for a high specialization of the installed hardware, software and trained personnel. The downside is that these data centers can only be used by Grid computing workloads. Often universities and funding agencies encourage research groups to share a common cluster installation and profit from the economies of scale effect. These cluster installations are mostly tailored to suit high performance computing (HPC) workloads, which require a tight interconnect between individual worker nodes in the cluster.

Most shared HPC-centers have to provide an operating system which can be used by all involved research groups. With the Scientific Linux [2] distribution, the HEP community maintains a custom operating system to achieve reproducibility and standardization among the many participating Grid sites. Most experiment-specific software in use today run on a Scientific Linux installation to fulfill the requirements set by the experiment's collaborations.

The opportunities for HEP-research to profit from such shared installations exist and in light of the aforementioned computing challenges, every effort should be taken in order to have access to these resources.

Furthermore, commercial cloud offerings provide standardized APIs for automated booking of virtual machines and network configurations. By uploading customized virtual machine images, a wide range of software and workflow requirements can be met. Thus, this services are a viable option to acquire additional resources during peak hours, like conference preparations or for computation campaigns.

2. Technology Choices

The technologies, which have been chosen to integrate remote resources into HEP workflows described in this document, will be introduced in the following. Many similar or complimentary products exist on the market today and the selection made is based on various factors which will be listed with each product. For other use cases, one might arrive at a different selection, which might be better suited.

2.1. HTCondor

The open-source HTCondor project provides a workload management system which is highly configurable and modular [3]. 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 NAT portal, to the central HTCondor pool are available. The Connection Brokering (CCB) service is especially valuable to connect virtual machines to the central pool [3]. These features and the well-known ability of HTCondor to scale to O(100k) of parallel batch jobs lets us decide to use HTCondor as a workload management system.

2.2. ROCED

Besides a cloud site and a flexible batch system, a central component is required to dynamically manage the virtual machines depending on the demand for computing resources. For this purpose, the cloud meta-scheduler ROCED (**R**eponsive **O**n-demand **C**loud **E**nabled **D**eployment) has been developed at the IEKP since 2010 [4]. ROCED is written in a modular fashion and the interfaces to batch systems and cloud sites are implemented as modules. This makes ROCED independent of a specific user group or workflow. For the described use case, the HTCondor and a special HPC-cloud module were implemented. The ROCED source code is available on GitHub [5].

3. Job Submission and Workflow Management

An overview of the jobs submission and management of jobs and the virtual machine management is given in Figure 1. The user submits jobs to the central HTCondor instance which distributes the jobs automatically to available resources. Users can add a special HTCondor ClassAd flag to signal they need access to the local storage resources. In this case, jobs are only scheduled to local resources. Otherwise jobs can also be submitted to remote cloud sites. Apart from the possibility to request local storage resources, the user does not need to track where jobs are run as the same HTCondor tools can be used to monitor, manage and debug local and remote jobs.

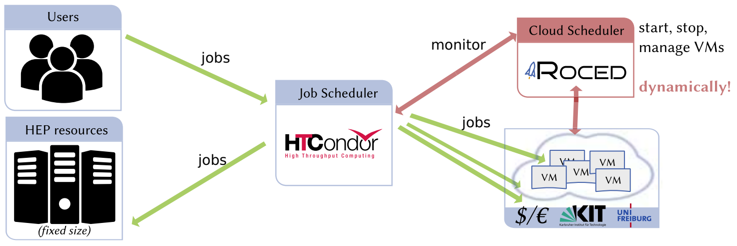

Figure 1. Schematic overview of the submission and management of jobs and virtual machines. The central component is HTCondor, which collects user jobs and distributes them to the available resources. The ROCED cloud scheduler component starts remote VMs on-demand.

4. Hybrid HPC System

The concept of a hybrid HPC system, which can run native batch jobs and jobs in a fully-virtualized environment, has been developed together with colleagues from the University of Freiburg [6]. Here, a HPC cluster with 16,000 cores is shared between three diverse user groups: elementary particle physics, neuroscience and microsystem engineering. Very early in the project it became clear that satisfying the software requirements for these three groups with the same software installation would be tough. While some groups rely on native access to the hardware, like InfiniBand for MPI, other groups, including our institute, require a dedicated operating system and a elaborate software stack. Instead of losing synergy effects by a static splitting of the cluster, we decided to implement a flexible virtualization approach. Here, the batch system of the cluster can either be used to start native jobs or to book virtual machines. These machines are managed via an OpenStack installation which runs alongside the batch client on each cluster node. Requesting OpenStack VMs via the scheduling process of the batch achieves a seamless integration of native and VM-based jobs.

The ROCED cloud scheduler has a module to request additional virtual machines on this shared HPC system every time the local resources are not sufficient to process queued jobs. This is a very dynamic procedure and depends on the submission pattern of the institute users. As soon as a virtual machine has been booted it connects to the central HTCondor instance and a job is started within seconds. Once the job has been completed, the virtual machine continues to accept and process job from HTCondor. If no additional jobs are available, the machine will automatically shut-down after a predefined wait period of some minutes.

Figure 2 shows a period of 36 days of HPC cluster usage. Up to 9000 virtual cores were used at the same time and filled by user jobs.

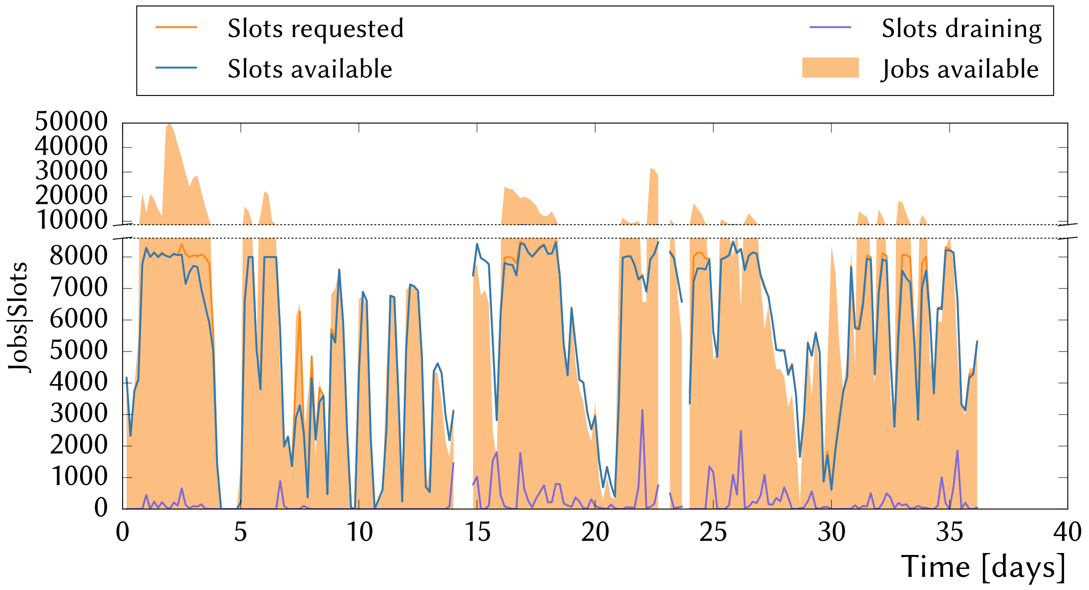

Figure 2. Utilization of the shared HPC system by booted virtual machines. Up to 9000 virtual cores were in use at peak times. The fluctuations in the utilization reflects the patterns of the submission of jobs by our institute users. The number of draining slots displays the amount of job slots still processing jobs while the rest of the node's slot are already empty.

5. Local Opportunistic Resources with Docker

In addition to use remote HPC or cloud resources, also the local institute's computing resources can be used in a more optimal way. Desktop machines today are quite powerful dual-core or even quad-core systems and are often idle with only the browser or email client in use. Our institute has more than 150 desktop CPU cores with sufficient RAM which can be used to process computation jobs during idle time of machines and during the night phases and the weekend. In order to provide our users with a pleasant and modern desktop experience, we employ an up-to-date Ubuntu installation on our desktop machines, while most of our experiment software requires a Scientific Linux installation.

In the past, we employed OpenStack to host virtual machines with the proper OS. However we recently moved to using the Docker container solution in conjunction with HTCondor as this provides a more lightweight solution, both in administrative work and resource consumption. A recent HTCondor startd client is able to start a selected Docker container automatically on a compute host.

Using this technique, Docker containers, which provide the exact same system libraries and binaries as if the jobs were running within a native Scientific Linux, are used for the user job execution. The required HTCondor settings to start the Docker container are automatically added to a user job if it gets scheduled to one of the Docker desktop nodes.

6. Commercial Cloud Offerings from 1&1 Internet SE

Commercial cloud provides are an alternative way to cushion peak demand. The 1&1 Internet SE is one of Europe's leading internet service provider with a strong global presence. Jointly, the 1&1 and the KIT teams executed a pilot project to evaluate the Cloud Server product for HEP jobs. The Cloud Server product offers dynamic provisioning of VMs and accurate billing depending on the machines uptime and configuration.

A backend to ROCED for the Cloud Server API [7] was developed to enable the demand-driven scheduling of VMs. We uploaded a custom VM-Image with Scientic Linux 6.7, CVMFS and HTCondor support to the 1&1 Cloud Server and also setup a dedicated CVMFS squid proxy machine which gets automatically booted by ROCED if any worker nodes are running.

We took advantage of the fact that the load distribution is lower during night time due to the typical customer profile of 1&1. This allows to boot many cores without competing with or affecting other customers of 1&1.

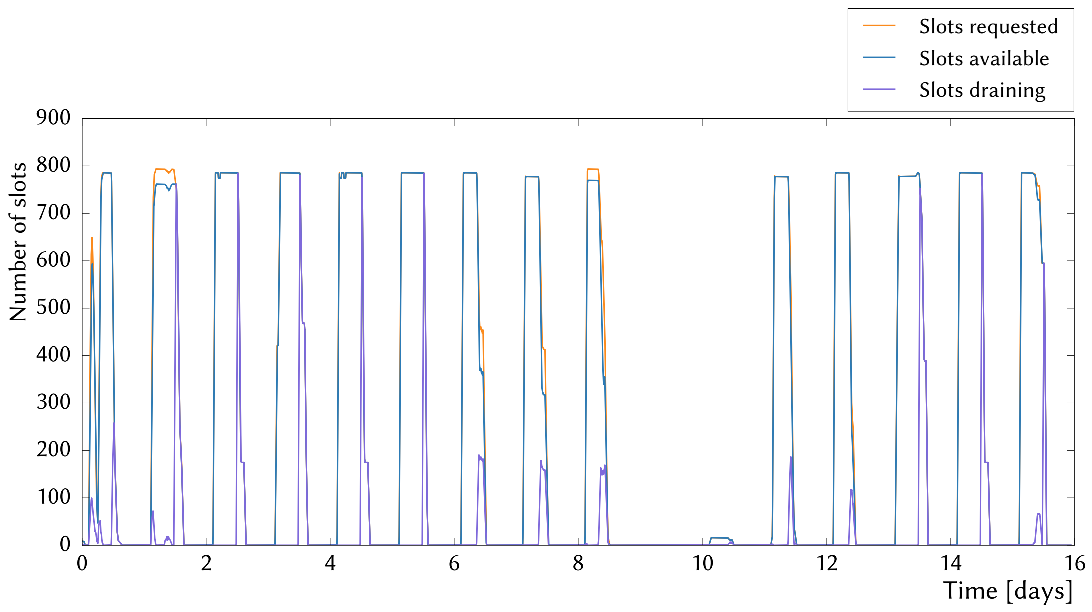

Figure 3. Utilization plot of the 1&1 Cloud Server offering over a period of 16 days. ROCED was configured to boot up to 800 virtual cores, if there is sufficient demand, and the operating period was limited to the night hours.

Figure 3 shows the resource utilization over a period of 16 days. ROCED was configured to only boot virtual machines during the night time and a maximum number of 800 cores. As visible in the Figure, the resources were fully utilized almost every night with user jobs. In nights were not all cores were provisioned, no user jobs were submitted to HTCondor or all user jobs could be handled by local resources.

This test phase proved that jobs of users from multiple experiments run reliably on the remote and virtual worker nodes and API-based scheduling without manual intervention is possible.

7. Conclusion

Our institute runs a flexible computing system which is able to leverage resources from multiple sources: a shared HPC System using virtualization with OpenStack, local opportunistic resources with Docker and commerical cloud offerings, for example the 1&1 Internet Cloud Server. This mix provides the institutes's users with additional, on-demand resources in peak hours. Using HTCondor for job submission and management and ROCED for provisioning of remote resources, this integration is transparent to the users. Since we started to use external resources in our production batch farm, more than half a million CPU hours have been provided by remote sites so far.

This dynamic resource management model gradually replaces our institute's private HEP-only cluster: both in ease of use and overall computing capacity.

References

[1] OpenStack website `https://www.openstack.org` (17/02/2016)
[2] ScientificLinux website `https://www.scientificlinux.org/` (17/02/2016)
[3] HTCondor website `http://research.cs.wisc.edu/htcondor` (17/02/2016)
[4] Hauth T, Quast G, Kunze M, Büge V, Scheurer A and Baun C 2011 *Journal of Physics: Conference Series* **331** 062034 Dynamic Extensions of Batch Systems with Cloud Resources URL `http://stacks.iop.org/1742-6596/331/i=6/a=062034`
[5] Erli G, Fleig G, Hauth T and Riedel S Roced cloud meta-scheduler project website `https://github.com/roced-scheduler/ROCED` (17/02/2016)
[6] Meier K, Fleig G, Hauth T, Janczyk M, Quast G, von Suchodoletz D and Wiebelt B 2016 *Journal of Physics: Conference Series* **762** 012012 Dynamic provisioning of a HEP computing infrastructure on a shared hybrid HPC system URL `http://stacks.iop.org/1742-6596/762/i=1/a=012012`
[7] 1&1 Cloud Server SDK Python `https://github.com/1and1/oneandone-cloudserver-sdk-python` (29.9.2016)