Experience with dynamic resource provisioning of the CMS online cluster using a cloud overlay

Jean-Marc Andre[8], *Ulf* Behrens[4], *James* Branson[1], *Philipp* Brummer[3,11], *Olivier* Chaze[3], *Sergio* Cittolin[1], *Diego* da Silva Gomes[3,*], *Georgiana-Lavinia* Darlea[6], *Christian* Deldicque[3], *Zeynep* Demiragli[6], *Marc* Dobson[3], *Nicolas* Doualot[8], *Samim* Erhan[5], *Jonathan Richard* Fulcher[3], *Dominique* Gigi[3], *Maciej* Gladki[3], *Frank* Glege[3], *Guillelmo* Gomez-Ceballos[6], *Jeroen* Hegeman[3], *Andre* Holzner[1], *Michael* Lettrich[3], *Audrius* Mecionis[8,9], *Frans* Meijers[3], *Emilio* Meschi[3], *Remigius K.* Mommsen[8], *Srecko* Morovic[8], *Vivian* O'Dell[8], *Luciano* Orsini[3], *Ioannis* Papakrivopoulos[7], *Christoph* Paus[6], *Andrea* Petrucci[2], *Marco* Pieri[1], *Dinyar* Rabady[3], *Attila* Racz[3], *Valdas* Rapsevicius[8,9], *Thomas* Reis[3], *Hannes* Sakulin[3], *Christoph* Schwick[3], *Dainius* Simelevicius[3,9], *Mantas* Stankevicius[8,9], *Cristina* Vazquez Velez[3], *Christian* Wernet[3], and *Petr* Zejdl[8,10]

[1]University of California San Diego, San Diego, USA
[2]Rice University, Houston, USA
[3]CERN, Geneva, Switzerland
[4]Deutsches Elektronen-Syncrotron, Hamburg, Germany
[5]University of California Los Angeles, Los Angeles, USA
[6]Massachusetts Institute of Technology, Cambridge, USA
[7]National Technical University of Athens, Athens, Greece
[8]Fermi National Accelerator Laboratory, Batavia, USA
[9]Also at Vilnius University, Vilnius, Lithuania
[10]Also at CERN, Geneva, Switzerland
[11]Also at Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract. The primary goal of the online cluster of the Compact Muon Solenoid (CMS) experiment at the Large Hadron Collider (LHC) is to build event data from the detector and to select interesting collisions in the High Level Trigger (HLT) farm for offline storage. With more than 1500 nodes and a capacity of about 850 kHEPSpecInt06, the HLT machines represent similar computing capacity of all the CMS Tier1 Grid sites together. Moreover, it is currently connected to the CERN IT datacenter via a dedicated 160 Gbps network connection and hence can access the remote EOS based storage with a high bandwidth. In the last few years, a cloud overlay based on OpenStack has been commissioned to use these resources for the WLCG when they are not needed for data taking. This online cloud facility was designed for parasitic use of the HLT, which must never interfere with its primary function as part of the DAQ system. It also allows to abstract from the different types of machines and their underlying segmented networks. During the LHC technical stop periods, the HLT cloud is set to its static mode of operation where it acts like other grid facilities. The online cloud was also extended to make dynamic use of resources during periods between LHC fills. These periods are a-priori unscheduled and of undetermined length, typically of several hours, once or more a day. For that, it dynamically follows LHC beam states and hibernates Virtual Machines (VM) accordingly. Finally, this work presents the design and implementation of a mechanism to dynamically ramp up VMs when the DAQ load on the HLT reduces towards the end of the fill.

* Corresponding author: diego@cern.ch

1 Introduction

Deployed on the facility for the Compact Muon Solenoid experiment (CMS) at the Large Hadron Collider (LHC), the CMS Online Cluster provides the computing power needed for the data acquisition and selection of interesting collision events for later offline storage and analysis. The High Level Trigger (HLT) farm is the biggest resource of this cluster, having more than 1500 nodes and summing up to 850 kHEPSpecInt06 of CPU capacity, for a total of 37k physical cores (74k virtual cores if counting hyper-threads). This is comparable to the amount of processing resources provided by all the CMS Tier1 Grid sites together.

During periods of no data taking, like for instance during Technical Stops and Machine Development weeks, the HLT farm is a mostly idle resource. Moreover, even when the LHC is operating, there are shorter periods with no data between fills for physics. Such intervals last typically a few hours and are called interfills. The motivation for this work is then to repurpose the HLT farm for offline processing like other Grid sites when there's no data-taking (Fig. 1).

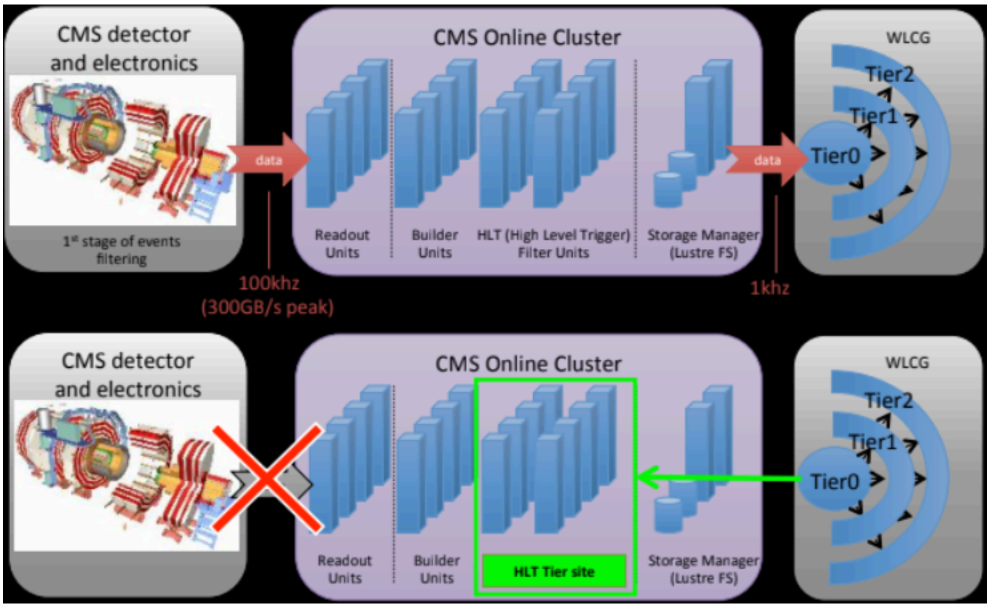
CMS detector and electronics
1st stage of events filtering
data
100khz (300GB/s peak)
CMS Online Cluster
Readout Units
Builder Units
HLT (High Level Trigger) Filter Units
Storage Manager (Lustre FS)
data
1khz
WLCG
Tier2
Tier1
Tier0
CMS detector and electronics
CMS Online Cluster
Readout Units
Builder Units
HLT Tier site
Storage Manager (Lustre FS)
WLCG
Tier2
Tier1
Tier0

Fig. 1. During normal data taking the CMS online cluster builds and selects events for processing by CMS grid sites (top). When no data is acquired, the HLT farm becomes a grid site (bottom).

For that to be possible, the following three major requirements must be achieved:

- **Isolation**: it should not interfere nor have any impact on the data taking;
- **Fast turnaround**: in order to profit from interfills as short as a couple of hours, the resources must be quickly allocated/deallocated for the Grid use;
- **Job resuming**: as the Grid execution of jobs must be interrupted when the next upcoming data-taking fill would start, some ability to pause grid jobs and resume them potentially hours later is needed.

On our previous work [1], we've shown how to build a cloud overlay on top of the HLT farm capable of running Grid jobs. It provided proper isolation but a solution for fast turnaround and job resuming was still being designed. In this work we'll show how these requirements were finally addressed. After a commissioning phase, the HLT cloud has been in production for two years. As we gained operational experience with it, some updates were made and new features implemented. In this paper we'll present the current status of the overlay model and share our operational experience. Finally, we'll show the contribution it made to the CMS offline data processing.

2 The Online Cloud overlay

The Figure 2 below shows all the components involved for providing the cloud overlay.

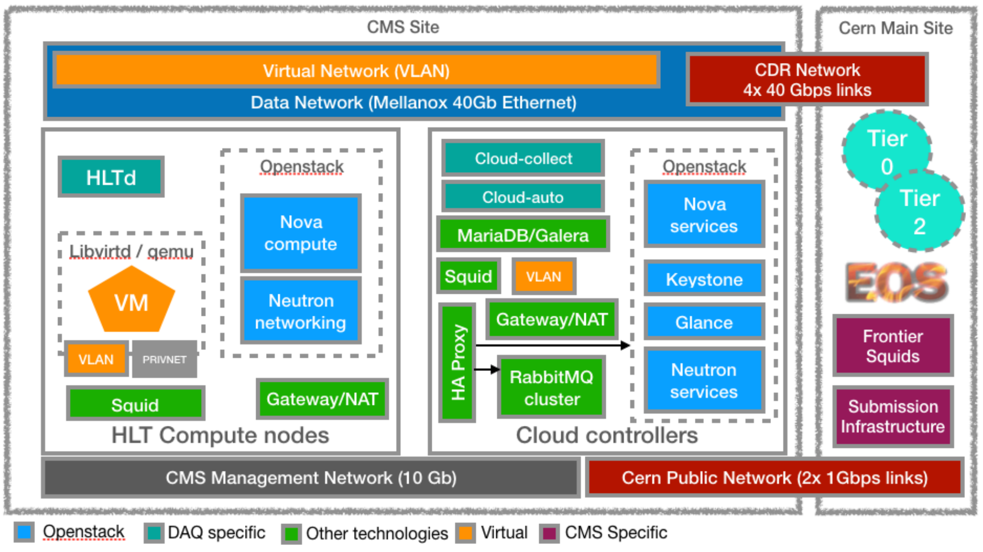

Fig. 2. Architecture for the Online Cloud overlay

In this architecture, there are two types of hosts: the HLT farm machines themselves, called HLT Compute nodes, and three servers running core services needed to drive the cloud, called Cloud Controllers. Two distinct networks connect them. The Management Network is a 10Gbps network providing basic connectivity to control every machine at the CMS Site, while the Data Network is a fast 40Gbps network used for the flow of the data acquired by the detector. 4x40Gbps links (160 Gbps in total) connect the data network to the CERN's Central Data Record network (CDR), thus allowing for fast transfer of data to/from the EOS storage instance for CMS at CERN. Different than the HLT Compute nodes, the Cloud Controllers also connect to a third network, the CERN's public network (GPN). This is done via 2x1Gbps links and is needed to reach outside services/hosts otherwise not reachable through CDR.

OpenStack [2] has been deployed to provide the virtualisation layer. Each HLT Compute node installs two services: Nova Compute and Neutron Networking. The former talks to the local Libvirt daemon to create virtual machines running Grid enabled software. The later uses the Neutron macvtap driver to create a virtual network interface used for the communication of VMs. On the Controller nodes, the OpenStack Nova services talk to the Nova Computes to manage the VMs and discover resources. Likewise, the central Neutron services deal with the management of the virtual network in the computing nodes by provisioning them IPs and basic routes. The Keystone service implements the OpenStack

Identity API needed for authentication and authorisation. Finally, Glance manages the images the VMs run.

On the CMS Online cluster, Frontier Squid [3] is used for the distribution of the detector conditions needed for the data taking. It is defined in an hierarchical manner that follows the distribution of the racks for scalability and fault tolerance. This same hierarchy was re-used by the cloud to get Grid job conditions from central Frontier servers at CERN and to provide CVMFS access for the VMs. However, at the top of the hierarchy requests are sent to the Cloud Controllers squids instead of the CMS Online ones. These Cloud Controllers squids therefore serve as a top level cache for the cloud. On the HLT nodes, a different Squid instance is used for the cloud to isolate it from the Squid usage by the data taking.

The virtual machines access two distinct networks. The Virtual Network is a tagged IEEE 802.1Q VLAN configured on top of the fast Data network. This VLAN not only provides traffic isolation for the VMs, but also deals with the network segmentation underneath: each VM has a flat view of the network when talking to each other and to the Cloud Controller gateways. The second network is for private communication between the VM and its hypervisor. It is used for NATing CDR traffic directly on the HLT Compute node to avoid concentrating it all on the few Cloud Controllers. Additionally, the VM uses this private channel to access the Squid instance on its hypervisor.

The Cloud Controllers also gather together to provide two more core services: the RabbitMQ [4] cluster and the MariaDB Galera [5] distributed database. RabbitMQ is the default messaging technology chosen by OpenStack services to provide loosely coupled communication. It has been configured with the *ha-all* (high availability) policy for the queues so that messages don't get lost in case of Controller node failures. MariaDB Galera is the SQL database OpenStack uses for keeping track of VMs, hypervisor states, allocated network IPs, etc. It has also been clustered on the Cloud Controllers with the intent of achieving DB fault tolerance, and run in Active/Passive mode.

HAProxy [6] runs on the Controller machines and listens for requests on two Virtual IPs (VIPs) that are dynamically assigned to the Controllers using Corosync/Pacemaker [7]. The Compute nodes and VMs only contact the Controllers via these VIPs, thus ensuring core OpenStack services and VLAN network routing remain highly available upon Controller failures. HAProxy then distributes the load through the various services running on the three Controllers for scalability. More Controllers can then be added should scalability limits be reached. Finally, CMS-DAQ specific daemons are used for switching HLT Compute nodes between two operating modes: plain DAQ and Cloud.

2.1 DAQ and Cloud daemons

While the architecture components explained in the previous section provide all that is needed for running Grid jobs in an isolated environment, the underlying computing resources would still be statically defined. The dynamic allocation of resources is however possible due to the interaction of three CMS-DAQ designed components.

The HLTd is the DAQ daemon in charge of coordinating the HLT processing work for the data taking. It is deployed on every HLT node and has been extended to provide an extra API for switching the operating mode. When in the default DAQ mode, it shuts down any OpenStack services on the node and makes sure no VMs are active before the machine joins a DAQ Run. In Cloud mode, any ongoing Run process is terminated and OpenStack services are brought back operational so that VMs can be managed.

The Cloud-collect is a simple daemon that deals with different monitoring sources to collect information such as the LHC Beam state, the CMS DAQ status, DAQ average CPU load, etc., and it stores this information on MariaDB. The consumer of this monitoring data is the Cloud-auto daemon, which is the main cloud daemon responsible for dynamically controlling the HLT resources. It does so by contacting HLTd on compute nodes to (de)allocate them depending on the identified running conditions and configurable Cloud operation modes.

2.2 VM hibernation and resuming

The Cloud-auto daemon leverages the fast turnaround of resources and job resuming by requesting the cloud VMs to hibernate (suspend) before returning them back to DAQ mode, and by resuming them later when the resource can be used opportunistically again.

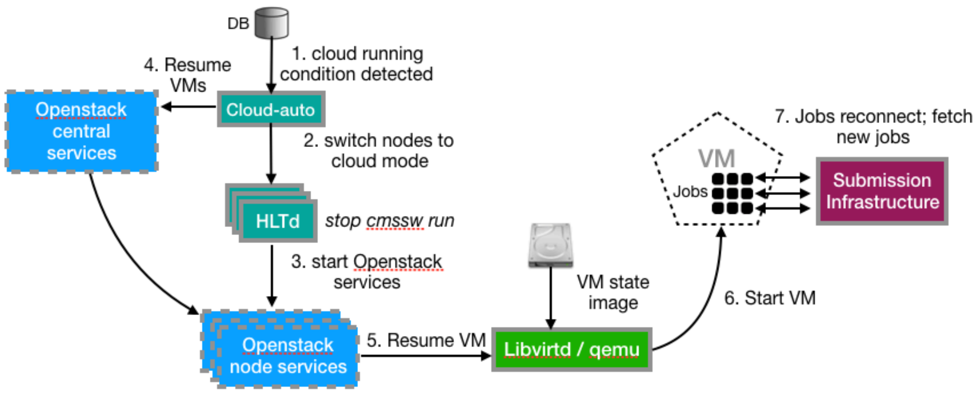
DB
1. cloud running condition detected
4. Resume VMs
Openstack central services
Cloud-auto
2. switch nodes to cloud mode
HLTd
stop cmssw run
3. start Openstack services
Openstack node services
5. Resume VM
Libvirtd / qemu
VM state image
6. Start VM
VM
Jobs
7. Jobs reconnect; fetch new jobs
Submission Infrastructure

Fig. 3. Interaction of the architecture components while resuming virtual machines for Cloud use

The Figure 3 above shows what typically happens when switching to Cloud mode. (1) The Cloud-auto identifies a running condition (i.e. identifies that there is no ongoing data taking); (2) it then contacts in parallel the HLTd running on every node and asks them to switch to Cloud mode; (3) after properly terminating the HLT processes, HLTd brings the OpenStack Nova compute and Neutron networking services up; (4) the Cloud-auto in turn requests all the VMs to resume using the OpenStack central services; (5) on the HLT node, OpenStack asks Libvirtd to resume the VM by loading its previous snapshot from disk; (6) the VM eventually resumes and thus the jobs continue running from where they left (no checkpoint mechanism is needed); (7) finally, the job slot eventually reconnects to the central Submission Infrastructure to provide updates or to eventually fetch a new job.

Whenever the running condition finishes, similar steps are taken, but in the opposite direction so that VMs get hibernated. Such switching cycles take less than 20min to complete. About 4 minutes are needed to contact and wait the HLTd to switch the HLT nodes to/from Cloud node, and resuming/suspending the VMs take up to 16min depending on the hardware type. However, for most flavours of machines this operation takes around 10 minutes, being essentially driven by the time needed to load/dump the VM memory snapshot from/to disk.

Resuming jobs potentially hours later this way is however only possible due to a feature of the HLT cloud. On the central Submission Infrastructure, the HLT Grid site has a special configuration for the job time-outs. Instead of the default 20 minutes, the job slots are only considered dead after 24 hours. Typical fills of physics on the LHC last around 9 hours, but on special cases it can take much longer, though rare to reach 24h. Nevertheless, more often it might happen the HLT farm is needed for running special DAQ tests when there is no data taking, and such tests can often last a day long.

2.3 Cloud operation modes

The Cloud-auto can be configured to drive the cloud in different operation modes:

- **Interfill**: in this mode, it opportunistically runs the cloud on periods where there is no fill for physics (Fig 4). For that it follows the LHC Machine state to identify maintenance periods, and during normal operations it follows the Beam states to detect interfills. This is the most used operation mode.

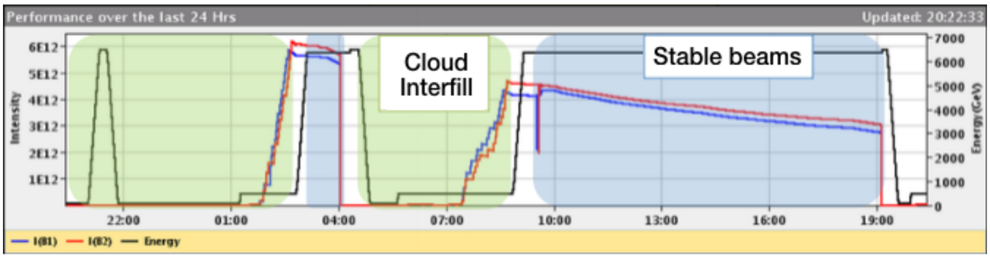

Fig. 4. Beam intensity (red and blue lines) and energy (black line) graph for a 24 hour period. The green areas are examples of interfill periods.

- **Fill**: this mode detects stable conditions during the fills where not all the HLT computing power would be needed, and progressively ramps up resources for cloud use. Fig. 5 demonstrates the potential of resources that can be leveraged in this mode. Typically, the higher CPU capacity is only needed at the beginning of the fill, lowering as it approaches the end. However, various conditions other than the HLT CPU load are also involved until it is considered safe enough for ramping cloud resources: DAQ state, trigger rate, ramdisk occupancy, and others. This operation mode was commissioned over the course of the LHC Run2.

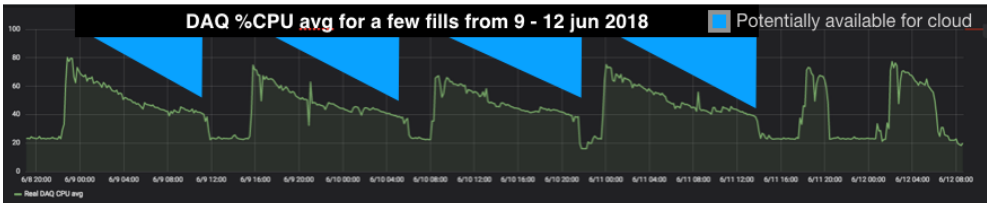

Fig. 5. During stable conditions, a fraction of resources can be allocated for Cloud use.

- **Fill + Interfill**: the combination of both modes.
- **On/off**: in this mode the cloud is switched on/off regardless of any condition.

3 Contribution to CMS offline processing

3.1 Results

The CMS Online cloud joined the CMS Offline production in the second half of 2016 after a few months of commissioning. During 2017, it provided up to 40k virtual cores for the collaboration. With those resources it managed to run 10 million jobs successfully (Fig 6), which represented 17% of the total Tier1s successful jobs.

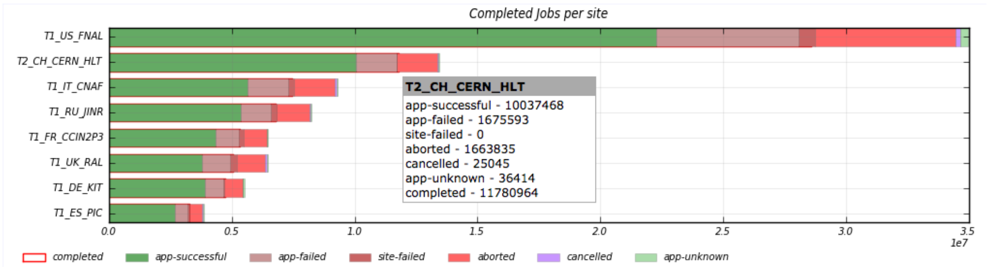

Fig. 6. Completed jobs during 2017.

In the first half of 2018, it made available up to 63k virtual cores and reached 7 million successful jobs (Fig 7), which was more than any other Tier1 site during the same time, or 30% of the total. Despite of the challenge of exploiting HLT opportunistically, the failure rate was comparable to that of the other Tier1s since it started operating.

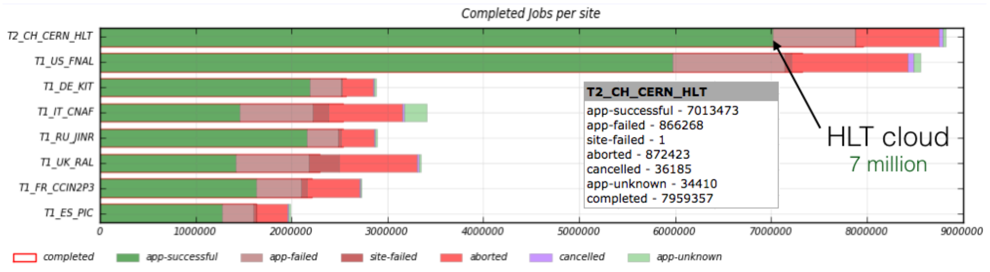

Fig. 7. Completed jobs, first half of 2018.

3.2 Known difficulties from experience

After the two years of operational experience, the following encountered difficulties are considered the most relevant ones:

- **Adapting to DAQ infrastructure changes**: since the Online Cloud does parasitic use of the DAQ-HLT resources, it must be provided on top of what is in use for the data taking for what concerns the Operating System, Networking and Computing hardware. Therefore, changes made on the HLT need to be followed and could require reviewing, updating and re-validation of the cloud software as well. One particular disrupting example was when the HLT farm migrated from Scientific Linux CERN 6 to CentOS CERN 7. Such change required the cloud updating from the OpenStack Grizzly version to Ocata, but the deprecation of important features and components like the Nova-networking implied in the whole cloud overlay to be re-worked. Another example, the addition of a significant amount of new machines to HLT required not only new scalability tests, but also the tuning of thresholds used to kill the cloud VMs upon monitored critical conditions.

- **Making efficient use of resources**: both the LHC machine and the CMS detector status and plans need to be tracked for the most suitable cloud operation mode to be set. Failing to do so may imply inefficient use of the resources. For instance, during low energy LHC runs a large fraction of resources can be temporarily assigned for dedicated cloud use. Conversely, ramping Beam energy quickly would not give the cloud enough time to hibernate the VMs and thus watching for interfills would be inappropriate. Another example, if DAQ would need the HLT for special tests potentially longer than 24h, the cloud VMs must be set to drain the jobs with some advance to avoid paused jobs eventually timing out. Finally, some other LHC/CMS conditions may be hard to anticipate and thus not reacting accordingly would also impact on the efficiency.
- **Troubleshooting job failures**: although the current monitoring tools help a lot spotting known issues, so far unknown problems often require analysis of the log and state of the VMs. However, these VMs are often only available for a few hours a day (i.e. during the interfills), not necessarily happening during working hours or when the issue is happening. Moreover, on some cases the problem may result in VMs getting killed due to reaching critical conditions, making it impossible to access the VMs for investigation.

4 Future work

Updating OpenStack from Ocata to Rocky (or newer) release is foreseen to happen during the LHC Long Shutdown 2 (LS2: 2019 - 2020), but no major change to the cloud architecture is expected to be required. This should bring bug fixes and improvements, but most importantly will catch up with DAQ required updates. For the monitoring, the Elastic Stack [8] (Kibana, ElasticSearch and Logstash/Beats) is expected to be deployed for the troubleshooting of job failures and log analysis. On the hardware side, an increase on the overall HLT capacity is also planned during LS2, with the older production HLT nodes becoming dedicated for cloud use. As this generation of dedicated machines has substantially more disk space (2 TB vs 250/500GB of current ones) we will evaluate the possibility of using them as a local buffer for un-merged job data. Finally, we will investigate the possibility of gathering some other types of compute resources across the Online Cluster.

References

1. Opportunistic usage of the CMS Online cluster using a cloud overlay: O. Chaze et al., International Symposium on Grids and Clouds 2016, Taipei, Taiwan
2. OpenStack: https://www.openstack.org/
3. Frontier Squid: http://frontier.cern.ch/
4. RabbitMQ message broker: https://www.rabbitmq.com/
5. MariaDB Galera: https://mariadb.com/kb/en/library/what-is-mariadb-galera-cluster/
6. HAProxy: http://www.haproxy.org/
7. Corosync/Pacemaker: https://clusterlabs.org/
8. Elastic Stack: https://www.elastic.co/products