EPJ Web of Conferences **214**, 03033 (2019) https://doi.org/10.1051/epjconf/201921403033
CHEP 2018

Evolution of HammerCloud to commission CERN Compute resources

Jaroslava Schovancová[1,*], *Alessandro* Di Girolamo[1], *Aristeidis* Fkiaras[1], and *Valentina* Mancinelli[1]

[1]CERN, 1 Esplanade Des Particules, Geneva, Switzerland

Abstract. HammerCloud is a testing service and framework to commission, run continuous tests or on-demand large-scale stress tests, and benchmark computing resources and components of various distributed systems with realistic full-chain experiment workflows.
HammerCloud, used by the ATLAS and CMS experiments in production, has been a useful service to commission both compute resources and various components of the complex distributed systems of the LHC experiments, as well as integral part of the monitoring suite that is essential for the computing operations of the experiments and their automation.
In this contribution we review recent developments of the HammerCloud service that allow use of HammerCloud infrastructure to test Data Centre resources in the early phases of the infrastructure and services commissioning process. One of the benefits we believe HammerCloud can provide is to be able to tune the commissioning of the new infrastructure, functional and also stress testing, as well as benchmarking with "standard candle" workflows, with experiment realistic workloads, that can be heavy for CPU, or I/O, or IOPS, or everything together. This extension of HammerCloud has been successfully used in CERN IT during the prototype phase of the "BEER" Batch on EOS (Evaluation of Resources) project, and is being integrated with the continuous integration/continuous deployment suite for Batch service VMs.

1 Introduction

The HammerCloud service is at the core of automation of computing operations. In WLCG [1] it is used by the ATLAS [2] and CMS [3] experiments, as well as CERN Compute, in order to commission, test, and benchmark computing resources and components of distributed systems with realistic full-chain experiment workflows. With HammerCloud we can run either functional tests, where we test the resources with a steady flow of test jobs, or stress tests, where we configure load intensity and observe behavior and functionality of a resource or a service under the load.

A great benefit of HammerCloud is the capability to test with full-chain experiment jobs, which perform the very same actions, utilize the very same environment, and access

---

* e-mail: jaroslava.schovancova@cern.ch

the very same services as standard physics analysis jobs. Therefore we can develop and configure tests in a way that points out infrastructure issues, avoiding transient issues with the user payload.

In ATLAS the HammerCloud service is used mainly in automation of ATLAS Distributed Computing operations, for functional testing and automatic exclusion and recovery of resources (more details in Fig. 1). We also perform benchmarking performance of resources with a standardized "standard candle" workflow. Another major contribution is in commissioning and integration of new resources, and commissioning of new components of distributed computing systems: e.g. pilot for ATLAS Workload Management System (WMS) PanDA [4], ATLAS Distributed Data Management (DDM) System Rucio [5], commissioning new data access protocols, etc. Lately, we introduced a new application that leverages HammerCloud to test services, in particular to test ATLAS Object Stores [6]. In order to successfully support the ATLAS testing needs, we collect static and dynamic information about the resources topology from the ATLAS Grid Information System (AGIS) [7].

In CMS the HammerCloud service is utilized to perform functional testing of resources, and commissioning of new resources as well as commissioning of new components of distributed computing systems. In order to configure CMS tests, we collect the topology information about the resources from the Computing Resources Information Catalog (CRIC) [8].

For the CERN batch system the HammerCloud service performs various tasks related to commissioning and improving utilization of resources: HammerCloud helped in commissioning of the BEER (Batch on Extra EOS Resources) project [9], and is at the core of the Continuous Integration / Continuous Deployment (CI/CD) tool to build images of virtual machines for the Batch service in CERN IT.

HammerCloud manages ca 80k jobs per day in ATLAS in ca 30 different tests per day, in CMS we run ca 39k jobs per day in ca 40 different tests per day, and within CERN batch testing we run between 150 and 750 jobs daily.

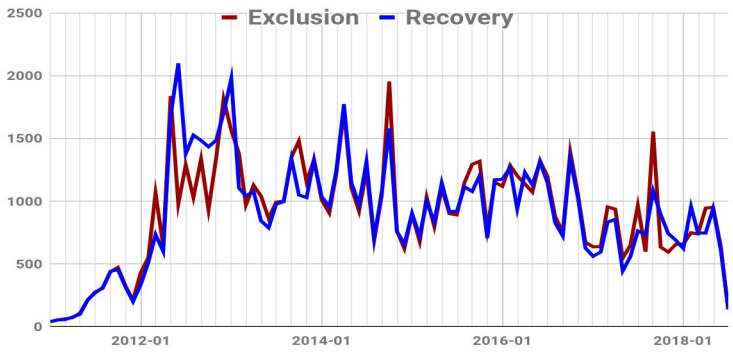

Fig. 1. ATLAS HammerCloud automatic exclusion and recovery actions. The plot shows number of automatic actions taken monthly by HammerCloud. The exclusion actions are triggered by functional test job failures, usually caused by an infrastructure issue. We aim to minimize exposure of ATLAS physicists to the infrastructure issues. Upon addressing the issue, when the tests start succeeding again, we automatically recover the resource, re-enabling the resource for physics analysis.

1.1 HammerCloud architecture and evolution

In the past 3 years we evolved HammerCloud in order to keep up with developments in the WMS and DDM systems of the experiments, to scale up for higher amounts of testing activity, to enable testing of 3rd party services functionality, and to make commissioning of resources even smoother. HammerCloud now supports a richer variety of workflows that the experiments run: e.g. from supporting only purely CPU-bound workflows, we are now able to support a variety of workflows with emphasis on different type of resource and different level of CPU-, memory-, and I/O-intensity. We can leverage an enriched variety of workflows to setup a testbed for WLCG Data Organization, Management, and Access [10] performance studies, to test ideas for the HL-LHC [11].

We evolved the HammerCloud submission core to support all the current testing needs, and it is flexible enough to adapt to future testing scenarios. We improved the resiliency of the core automation application.

The HammerCloud service is a Django [12] web framework application backed by a MySQL [13] database. We run HammerCloud infrastructure on CERN Agile Infrastructure resources [14]. To scale up and utilize our testing infrastructure, we integrated Celery cluster [15] backed by Redis [16] in our infrastructure. The HammerCloud schema is described in **Fig. 2**.

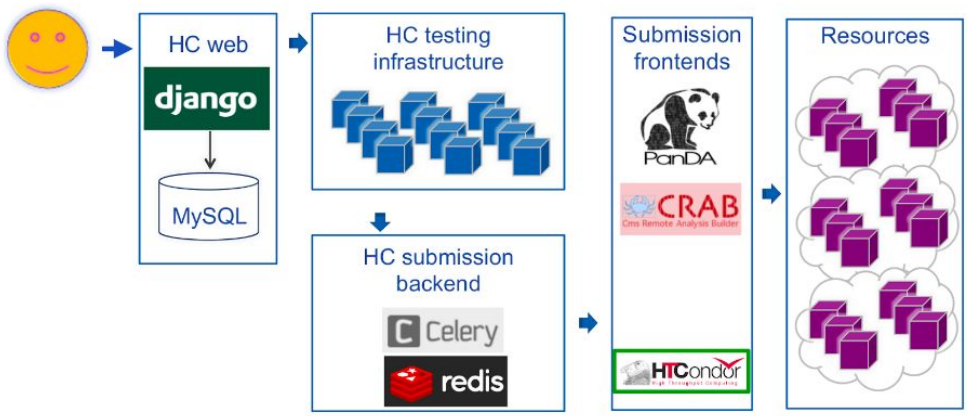

Fig. 2. HammerCloud schema overview. HammerCloud is a Django application, running on top of a MySQL database. The HammerCloud testing infrastructure runs on CERN Agile Infrastructure. The HammerCloud submission backend benefits from a Celery cluster backed by a Redis key-value store for the transport layer. The HammerCloud submission backend interacts with the submission frontends, the experiment WMS (PanDA, CRAB3 [17]) or the computing element (HTCondor-CE). The submission frontends interface to the resources where HammerCloud runs the tests.

2 HammerCloud for pre-commissioning of resources

While integrating new computing resources with our existing infrastructure, we often face many challenges with spotting, debugging, and addressing infrastructure and other issues. These can be categorized in a variety of ways, not limited only to networking (e.g. DNS name resolution, firewall issues, sudden changes in network bandwidth), issues with access to services and resources essential to run a job (database services and caches, Frontier server), convoluted with a multitude of ways that jobs can suffer (without any contribution from degraded infrastructure).

The goal we strive to achieve is to integrate heterogeneous resources transparently, in a way that minimizes exposure of our end users to infrastructure issues. Sometimes there are issues that go undetected, contributing to frustration among many teams. With over a decade of experience with resources commissioning, we developed an idea of how to make the commissioning process smoother: a standard HammerCloud job is a full-chain job that interacts with the distributed data management and workload management systems of the experiments, the resources/infrastructure, and a variety of services necessary for the successful job run. We would like to be able to reproduce this kind of environment, however, in a controlled way, where we can fail early, spot infrastructure issues and address them before exposing the user community to these issues.

We approached this challenge by choosing to swap the submission backend to WMS with a submission backend to the batch system (HTCondor cluster, or a HTCondor-CE, in our case). In this way we can submit a generic payload that interacts with the infrastructure in the same way as the standard HammerCloud job, uncovering issues in a controlled environment, essentially pre-commissioning any resources for computing activities before we sign-off and pass them to the Experiment as “ready”.

The pre-commissioning approach is not limited only to batch resources, there are many entities with similar objectives with respect to commissioning:

- Batch resources: we can leverage the pre-commissioning approach for in-house resources as well as to integration of public clouds. In addition, we can pre-commission images of virtual machines (VMs; this use case is described in further detail in the subsection 2.1), or explore and commission new ways to improve utilization of resources (e.g. BEER project).
- Containers: we can commission container images and validate environment configuration.
- Services functional testing: we can probe status and functionality of services, as we did e.g. in the case of ATLAS Object Store testing.
- Issuing load of any kind: we can perform a front-end load testing, or a DB load testing.
- Commission components of complex distributed systems: e.g. a DDM client component commissioning, or validating releases of 3rd party packages and dependencies.

2.1 CERN batch Continuous Integration / Continuous Deployment suite

The CERN batch resources are provisioned mostly as OpenStack [18] VMs, while we are gaining experience with containerized deployment at scale. We operate resources with 300k+ CPU cores in 24k+ VMs organized in several HTCondor [19] clusters. The VMs are configured with Puppet [20].

The VM configuration starts with a base operating system (OS) image, which we enrich and configure with Puppet. Full initial configuration of a batch VM usually takes around 3 hours. That is not a negligible amount of time and potentially wastes CPU cycles at the scale we need, therefore we make VM snapshots as a batch VM image, and instantiate batch VMs from the batch VM images, spending 3 hours only once to create the image, and later much less time with the final configuration of all the freshly spawned VMs.

Further we describe a CI/CD suite that contributes to improving the utilization of available compute resources at CERN. Its schema is shown in Fig. **3.**

1. We use Packer [21] to build a batch VM image: from a base OS image we spawn a VM, configure it with Puppet, and remove the part of the image that are specific for the VM used for build (e.g. hostname, host certificates, etc.). Such a batch VM image will be subjected to testing to probe its functionality in the subsequent steps, to uncover possible issues with infrastructure due to configuration changes, or due to changes in configuration dependencies.
2. We instantiate a few VMs in an OpenStack cluster based on the new batch VM image, and plug them in a HTCondor cluster.
3. We start HammerCloud tests to pre-commission the fresh VMs. This gives us an opportunity to test functionality and spot issues with infrastructure, VM configuration, dependencies, and job environment, in a controlled way. When issues show up, we can reject the batch VM image, address the issues or report to the dependency maintainers, and restart the process later with a new VM image.
4. When our tests pass successfully, we can sign-off the VM image and use it at scale in our environment.

The CERN batch CI/CD suite helps us to spot incompatible changes in the dependencies, and our environment. The steps, as well as the whole chain, can be automated, therefore we can perform them regularly, e.g. daily, limiting the amount of changes being taken into consideration, and providing a very recent configuration of a batch VM, further minimizing the time needed to finalize configuration of a new VM built from a new batch VM. Consequently, we can use the spare CPU cycles that we didn't spend on VM configuration, to run jobs.

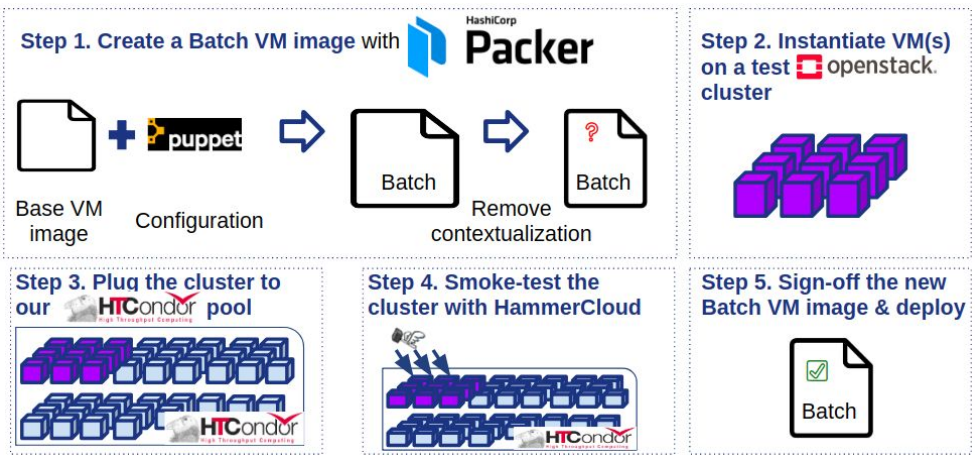

Fig. 3. Steps to create and validate a batch VM image, using the CERN batch CI/CD suite.

3 Conclusions

We have described the evolution of HammerCloud to accommodate a variety of workflows, from the standard experiment full-chain jobs submitted via workload management systems, through testing services, to submitting a generic payload to an HTCondor cluster.

The HammerCloud extension that allows a generic payload submission is integrated with the CERN batch CI/CD suite, and several other applications, such as commissioning releases of container images, as well as the ATLAS SW installation system releases, which can benefit from it.

To conclude, HammerCloud is a flexible service and a framework to provide means to test, commission, and benchmark computing resources or components of the distributed computing systems of today, as well as support studies to prepare for HL-LHC computing challenges.

References

1. *WLCG*. http://wlcg.web.cern.ch/ Accessed 25th October 2018.
2. ATLAS Collaboration. *The ATLAS Experiment at the CERN Large Hadron Collider*. JINST, 3:S08003 (2008)
3. CMS Collaboration. *The CMS Experiment at the CERN Large Hadron Collider*. JINST 3:S08004 (2008)
4. F. H. Barreiro Megino et al. *PanDA for ATLAS Distributed Computing in the next decade*. Journal of Physics: Conference Series, 898(5):052002 (2017)
5. M. Barisits, T. Beermann, V. Garonne, T. Javurek, M. Lassnig, C. Serfon, *The ATLAS Data Management System Rucio: Supporting LHC Run-2 and beyond*, ACAT, Seattle, 2017
6. P. Love, T. G. Hartland, B. Douglas, J. Schovancová, A. Dewhurst. *Object store characterisation for ATLAS distributed computing*, The 23rd International Conference on Computing in High Energy and Nuclear Physics, Sofia, 2018, these proceedings (2019)
7. A. Anisenkov, A. Di Girolamo, M. Alandes Pradillo. *AGIS: Integration of new technologies used in ATLAS Distributed Computing*. Journal of Physics: Conference Series, 898(9):092023 (2017)
8. A. Anisenkov et al. *Computing Resource Information Catalog: the ATLAS Grid Information system evolution for other communities*. The Nuclear Electronics and Computing, Montenegro, 2017
9. D. Smith et al. *Sharing server nodes for storage and compute*, The 23rd International Conference on Computing in High Energy and Nuclear Physics, Sofia, 2018, these proceedings (2019)
10. I. Bird, S. Campana, M. Girone, X. Espinal Curull, G. McCance, J. Schovancová. *Architecture and prototype of a WLCG data lake for HL-LHC*, The 23rd International Conference on Computing in High Energy and Nuclear Physics, Sofia, 2018, these proceedings (2019)

11. HEP Software Foundation. *A Roadmap for HEP Software and Computing R&D for the 2020s*. arXiv:1712.06982 (2017)
12. *Django web framework*. https://www.djangoproject.com/ Accessed 25th October 2018.
13. *MySQL database*. https://www.mysql.com/ Accessed 25th October 2018.
14. P. Andrade, T. Bell, J. van Eldik, G. McCance, B. Panzer-Steindel, M. Coelho dos Santos, S. Traylen, U. Schwickerath. *Review of CERN Data Centre Infrastructure*. Journal of Physics: Conference Series, 396(4):042002 (2012)
15. *Celery: Distributed Task Queue*. http://www.celeryproject.org/ Accessed 25th October 2018.
16. *Redis*. https://redis.io/ Accessed 25th October 2018.
17. M. Cinquilli et al. CRAB3: *Establishing a new generation of services for distributed analysis at CMS*. Technical Report CMS-CR-2012-139, CERN (2012)
18. *OpenStack*. https://www.openstack.org/ Accessed 25th October 2018.
19. D. Thain, T. Tannenbaum, M. Livny. *Distributed computing in practice: the Condor experience*. Concurrency - Practice and Experience, 17(2-4):323-356 (2005)
20. *Puppet*. https://puppet.com/ Accessed 25th October 2018.
21. *Packer by Hashicorp*. https://www.packer.io/ Accessed 25th October 2018.