Cloudscheduler: a VM provisioning system for a distributed compute cloud

Randall Sobie[1,2], *Frank* Berghaus[1], *Kevin* Casteels[1], *Colson* Driemel[1], *Marcus* Ebert[1], *Fernando* Galindo[3], *Colin* Leavett-Brown[1], *Danika* MacDonell[1], *Michael* Paterson[1], *Rolf* Seuster[1], *Shaelyn* Tolkamp[1], and *Jodie* Weldon[1]

[1]Department of Physics and Astronomy, University of Victoria, Victoria, British Columbia, Canada
[2]Institute of Particle Physics (Canada)
[3]TRIUMF, 4004 Wesbrook Mall, Vancouver, British Columbia, Canada

Abstract. We describe a high-throughput computing system for running jobs on public and private computing clouds using the HTCondor job scheduler and the cloudscheduler VM provisioning service. The distributed cloud computing system is designed to simultaneously use dedicated and opportunistic cloud resources at local and remote locations. It has been used for large scale production particle physics workloads for many years using thousands of cores on three continents. Cloudscheduler has been modernized to take advantage of new software designs, improved operating system capabilities and support packages. The result is a more resilient and scalable system, with expanded capabilities.

1 Introduction

In the field of particle physics, clouds are increasingly used to process and analyze data. It is possible that commercial clouds might satisfy the computational requirements of the next generation of global research projects; however, it is likely that dedicated storage facilities will be required to store and preserve the research data. One scenario may be a hybrid solution of dedicated and opportunistic, private and commercial compute resources, linked by high-speed networks to a distributed set of dedicated storage repositories.

This paper describes such a hybrid solution for running high-throughput computing workloads on compute clouds, irrespective of the underlying cloud software, location or its ownership. We call our design a *distributed compute cloud* where the resources are an aggregate of compute clouds hidden from the user.

The distributed compute cloud uses cloudscheduler for VM provisioning and scheduling, and HTCondor for job scheduling. Briefly, cloudscheduler reviews the HTCondor job queue and cloud resources to determine whether there are clouds that can start VMs that meet the job requirements. If a match is found, then cloudscheduler requests that the appropriate VM image be booted on the cloud. Once the VM is instantiated, it joins the HTCondor pool and the user job is sent to the running VM instance.

The original design of cloudscheduler was conceived in 2009 [1] and is based on ideas discussed in a paper describing "sky computing" [2]. The distributed compute cloud has run many millions of jobs on three continents for the ATLAS experiment at the CERN Laboratory

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).

in Geneva, Switzerland [3] and the Belle II experiment at the KEK Laboratory in Tsukuba, Japan [4].

Many of the custom and external software components have undergone significant change since the first version of cloudscheduler. Further, the years of production operation have provided insight on how to simplify and improve the system for usability, reliability and maintenance. In 2018, cloudscheduler was changed from a single Python code framework into a software platform with expanded functionality using current software technologies and practises. The new design is more robust, error tolerant and scalable, and is currently used for ATLAS and Belle II production workloads.

There are other strategies and software for utilizing clouds in particle physics. These include Vcycle, VMDIRAC and an extension of the HTCondor job scheduler. In Vcycle, the resource provider creates VMs for the experiments that draw jobs from the experiments' central queue of tasks [5]. VMDIRAC is a module for the DIRAC workload management system [6] that can start VMs on clouds [7, 8]. HTCondor can also be used to start VMs on Openstack clouds [9]. The distributed cloud computing system using cloudscheduler provides an infrastructure that can run workloads for any field or research without the need for application experts at each site. It is not dependent on project-specific workload management systems but can be integrated with them. Further, it can run workloads on all cloud types while the users or experiments only need to know about the familiar batch system interface to which they submit jobs.

2 Cloudscheduler

2.1 Overview

The architecture of the distributed compute cloud using cloudscheduler has been described in a number of papers [10, 11]. The key components are the HTCondor job scheduler, the cloudscheduler VM provisioning service and the compute clouds. HTCondor was selected as the job scheduler as it was designed to be a cycle scavenger [12], making it an ideal fit for a dynamic cloud environment where resources (VM instances) appear and disappear.

Cloudscheduler was redesigned over the past few years to improve reliability and scalability with expanded capabilities. Cloudscheduler Version 2 (CSV2) is a framework of component processes written in Python 3 and built around the MariaDB (SQL) relational database [13]. There are processes for control, data gathering (pollers), user interfaces and for VM provisioning (see Fig. 1).

The database allows data retrieval from multiple sources, and correlates information between different subsystems to obtain different views of the system data. The data is organized in tables, with rows representing objects and columns representing the attributes of an object. There are tables for global and local configuration parameters. CSV2 also generates ephemeral data in dedicated tables for objects like jobs or virtual machines. The database also adds authorization features that protects the integrity of the data while allowing both local and remote access.

Cloudscheduler has a RESTful web user interface (UI) for administration, control, and detailed monitoring of the clouds and jobs. The UI can be accessed through either a graphical user interface (GUI) using web browsers or a command line interface (CLI) client (with extensive man pages); the GUI and CLI have nearly the same functional capabilities.

Cloudscheduler has *pollers* that fill the MariaDB with information from the clouds and HTCondor. New entries are added to multiple tables in the database for each new job and VM. The state information of existing entries are updated by the *pollers*, and obsolete information (for completed jobs/VMs) is removed.

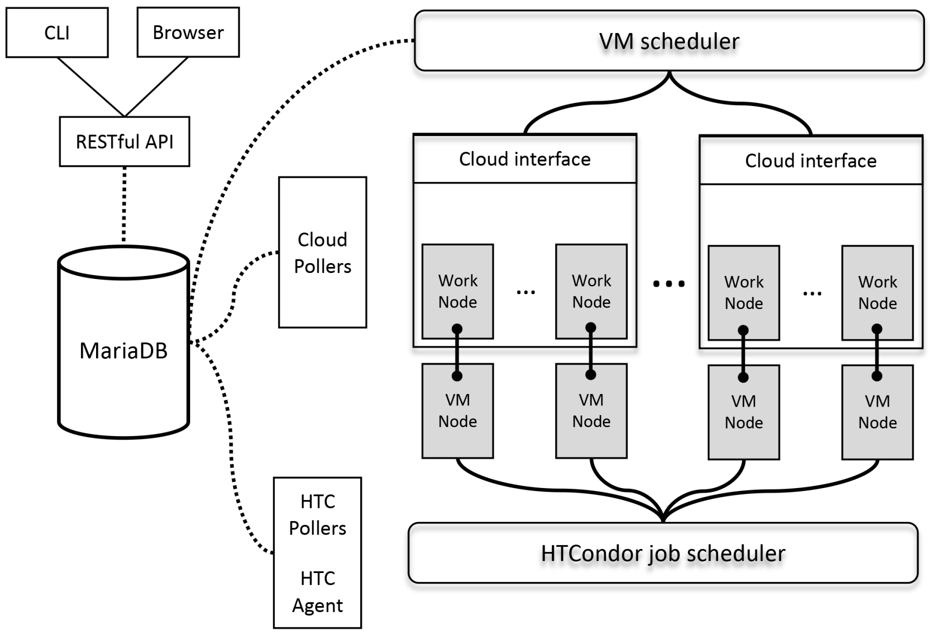

Figure 1. Overview of the cloudscheduler and HTCondor distributed compute cloud. The primary change is the introduction of the MariaDB for keeping track of the state of the system and "pollers" to gather information from the clouds and the job scheduler. The information in the database is used for scheduling, management and monitoring.

Fig. 1 shows a single *cloud poller*; however, there are separate *cloud pollers* for each cloud-type[1] (e.g. one poller can support multiple Openstack clouds). Fig. 1 also shows pollers for HTCondor (labelled as *HTC pollers*). There is a *HTC machine poller* that gathers information on the HTCondor machines and a *HTC job poller* that gathers information on the jobs in the HTCondor queue. The pollers consist of multiple tasks that run at different frequencies. For example, the *cloud poller* updates the MariaDB with information on the VMs every 60 seconds whereas it updates the list of VM images every 5 minutes. Typically, the dynamic information is updated every minute and the relatively static information on a longer timescale (between 5 minutes and one day).

Although Fig. 1 shows only one HTCondor instance, CSV2 can support multiple instances. For example, separate HTCondor instances are used by ATLAS and Belle II experiments, in part, due to differing security requirements. All HTCondor instances are supported by a single set of HTCondor pollers. However, each instance of HTCondor requires a *HTC Agent* to provide the correct security context when issuing requests to the HTCondor client on the VM (e.g. requesting a VM to de-register from the HTCondor pool).

[1]Currently there is API support for Openstack and Amazon EC2 clouds (support for other cloud APIs will be added on demand)

2.2 Workflow

The workflow is determined by the configuration of the system, the job information and the state of the VMs. The state of the VMs is dynamically determined by the information gathered by the *cloud poller*, *HTC job poller* and *HTC machine poller*, and stored in the MariaDB.

If there is a job in the HTCondor queue, then the *HTC job poller* creates a new entry in the MariaDB with the job information. The *VM scheduler* periodically queries the MariaDB for queued jobs and determines if there is a cloud with free resources that meet the requirements of the queued jobs. If the *VM scheduler* finds a cloud that is matched to the job(s), then it issues a request to start a VM (via the cloud API) to the cloud and creates an entry in the MariaDB for each new VM.

The contextualization of a VM is controlled by metadata that is passed to the "cloud-init" process [14]. There are a number of obligatory items to configure such as the HTCondor client. Cloudscheduler provides default metadata files to manage all obligatory items (stored in the MariaDB). The ATLAS and Belle II software environments are configured during the contextualization (see next section).

Once the VM has completed its contextualization, it is registered as a HTCondor machine. The *HTC machine poller* periodically retrieves the ClassAds of all the HTCondor machines via the HTCondor Python API and stores its ClassAd in the MariaDB.

If a VM remains idle for an extended period of time after completing one or more jobs, then either the job queue is empty or there are no queued jobs that can run in the VM. In this situation, the *VM scheduler* sets the retire-flag in the database entry of the VM. The VM is sent a message so that the HTCondor master daemon on the VM disables all the partitions, preventing them from accepting new jobs. The remaining jobs on the VM are allowed to complete and then the cloud is requested to shut down that VM instance.

2.3 Configuration and contextualization

Cloudscheduler can be configured from the CLI or GUI. The users can add or remove clouds, define VM instance types or flavours on each cloud, and specify quotas for each cloud. The VM images and SSH keys can be uploaded by logging in directly to the cloud or via the GUI. The GUI is accessed through a web browser and can also generate time series plots for each variable (stored in an InfluxDB). There are other configuration settings for the management of VMs with default parameters.

The contextualization of the VM instances for the ATLAS and Belle II experiments is very similar. Both use the micro-CernVM virtual machine image [15], the CernVM File System (CVMFS) [16], and optionally, *Shoal*, a dynamic web cache locator service [17]. The micro-CernVM image (approximately 20 MB) saves storage space and can be started instantaneously. The image contains a read-only Linux kernel and a CernVM File System (CVMFS) client. An array of squid proxy caches around the world are used to host frequently used files from CVMFS [18]. The CVMFS client is configured to use a specific squid cache; however, we can configure the CVMFS client to use the nearest squid cache to the VM by querying Shoal.

The contextualization process then sets up the HTCondor client, and the environment for the experiment. The accounting and monitoring processes are activated, and the fast (DB12) CPU benchmark suite [19] is run on the VM. The necessary host certificates and SSH keys are added to the VM.

2.4 Operations and status

The distributed compute cloud, using the new version of cloudscheduler, started production use in early 2019 and is integrated into the WLCG grid infrastructure [20]. It uses clouds in North America and Europe running mainly Openstack software, though Open Nebula, Amazon EC2, Google GCE and Microsoft Azure are also used.

The ATLAS production using the clouds in North America was shifted to the new version and tested extensively for many months. The Belle II production system was transitioned to the new version in June 2019. ATLAS and Belle II are now supported by a single instance of cloudscheduler but continue to use separate instances of HTCondor that are linked to their respective workload management systems.

The HTCondor instance dedicated to the ATLAS experiment is linked to the PanDA workload management system [21]. ATLAS typically uses a few hundred thousand cores at a given moment, and the distributed compute cloud contributes approximately 1% of the resources, comparable to the other Tier-2 compute centres operated in Canada.

The Belle II experiment uses the DIRAC workload management system [6]. We operate three DIRAC Site Directors in Victoria that submit pilot jobs to the HTCondor instance if there are queued jobs in the central DIRAC server at KEK. The distributed compute cloud has provided 13% of the resources (typically a few thousand compute cores) of the experiment since January 2018.

An important new feature is the ability to manage the jobs and clouds of multiple groups (experiments). This makes it possible for one experiment to opportunistically utilize the resources of another experiment whose resources are idle. Cloudscheduler can now be configured so that the opportunistic usage can be some fraction or all of the idle resources. If the resources are required, then the opportunistic usage is automatically scaled down by retiring those resources. The running jobs are allowed to complete so that the transition back to the fair share can take a number of hours. This feature has significantly improved the utilization of our resources.

In addition, the distributed compute cloud is using the Dynafed data federation service, developed by CERN IT, for locating input data [22, 23]. Dynafed federates existing storage systems and provides a link to the closest copy of the data on the basis of GeoIP information. Dynafed is used to federate existing storage of the ATLAS and Belle II experiments [24–26].

3 Summary

We have presented the design of a distributed cloud computing system based on the cloudscheduler VM provisioning platform. It remains a novel system for utilizing high-throughput workloads on multiple dedicated and opportunistic clouds located at remote locations and owned by other organizations. The system has provided significant resources to the ATLAS and Belle II particle physics experiments. The cloudscheduler VM provisioning system has been updated to current software standards to make it more scalable and reliable, as well as adding new functionality. It is currently planned to use cloudscheduler for the computing part of the proposed Belle II Canadian Raw Data Centre. Other plans include further integration and use of the Dynafed data federator to expand the running of data-intensive applications.

The cloudscheduler software is stored in GitHub (*https://github.com/hep-gc/cloudscheduler*) and documentation is found at *cloudscheduler.readthedocs.io*.

Acknowledgements
We would like to thank the support of the Natural Sciences and Engineering Research Council

of Canada and the Canadian Foundation of Innovation. In addition, the resources and generous support provided by Compute Canada, the Chameleon Project, the Edinburgh Compute and Data Facility (ECDF), the Leibniz Supercomputing Centre, the Institute of High Energy Physics (Austria), Amazon and Microsoft are acknowledged.

References

[1] P. Armstrong *et. al.*, Cloudscheduler: a resource manager for distributed compute clouds, arXiv preprint arXiv:1007.0050 (2010)

[2] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes, Sky computing, Internet Computing, IEEE, 13(5):43–51, 2009, doi:10.1109/MIC.2009.94

[3] R.P. Taylor *et. al.*, Consolidation of cloud computing in ATLAS, J. Phys.: Conf. Ser. 898 052008 (2017), doi:10.1088/1742-6596/898/5/052008

[4] R.J. Sobie *et. al.*, Utilizing clouds for Belle II J. Phys.: Conf. Ser. 664 022037 (2015), doi:10.1088/1742-6596/664/2/022037

[5] A. McNab *et. al.*, Managing virtual machines with Vac and Vcycle, J. Phys.: Conf. Ser. 664 022031 (2015), doi:10.1088/1742-6596/664/2/022031

[6] A Tsaregorodtsev *et. al.*, DIRAC Distributed Computing Services, J. Phys.: Conf. Ser. 513 032096 (2014), doi:10.1088/1742-6596/513/3/032096

[7] R. Grzymkowski *et. al.*, Belle II public and private cloud management in VMDIRAC, J. Phys.: Conf. Ser. 664 022021 (2015), doi:10.1088/1742-6596/664/2/022021

[8] A. Amoroso *et. al.*, A modular (almost) automatic set-up for elastic multi-tenants cloud (micro)infrastructures, J. Phys.: Conf. Ser. 898 (2017) 082031, doi:10.1088/1742-6596/898/8/082031

[9] B. Bockelman *et. al.*, Interfacing HTCondor-CE with OpenStack, J. Phys.: Conf. Ser. 898 092021 (2017), doi:10.1088/1742-6596/898/9/092021

[10] K. Fransham *et. al.*, Research computing in a distributed cloud environment, J. Phys.: Conf. Ser. 256 (2010) 012003, doi:10.1088/1742-6596/256/1/012003

[11] I. Gable *et. al.*, A batch system for HEP applications in a distributed IaaS cloud, J. Phys.: Conf. Ser. 331 062110 (2011), doi:10.1088/1742-6596/331/6/062010

[12] D. Thain, T. Tannenbaum and M. Livny, Distributed computing in practice: the Condor experience, Concurrency Computat.: Pract. Exper.2005;17:323 (2005), doi:10.1002/cpe.938

[13] MariaDB open-source relational database, https://mariadb.org

[14] Cloud_init: the standard for customizing cloud instances. https://cloud-init.io

[15] J. Blomer *et. al.*, Micro-CernVM: slashing the cost of building and deploying virtual machines, J. Phys.: Conf. Ser. 513 (2014) 032009, doi:10.1088/1742-6596/513/3/032009

[16] J. Blomer *et. al.*, New directions in the CernVM file system, J. Phys.: Conf. Ser. 898 062031 (2017), doi:10.1088/1742-6596/898/6/062031

[17] I. Gable *et. al.*, Dynamic web cache publishing for IaaS clouds using Shoal, J. Phys.: Conf. Ser. 513 032035 (2014), doi:10.1088/1742-6596/513/3/032035

[18] Squid is a caching proxy for the Web. http://www.squid-cache.org.

[19] P. Charpentier, Benchmarking worker nodes using LHCb productions and comparing with HEPSpec06, J. Phys.: Conf. Ser. 898 082011, doi:10.1088/1742-6596/898/8/082011

[20] The Worldwide LHC Computing Grid, http://wlcg.web.cern.ch

[21] F.H. Barreiro Megino *et. al.*, PanDA for ATLAS distributed computing in the next decade J. Phys.: Conf. Ser. 898 052002 (2017), doi:10.1088/1742-6596/898/5/052002

[22] Dynafed - The Dynamic Federation project. http://lcgdm.web.cern.ch/dynafed-dynamic-federation-project

[23] F. Furano, O. Keeble and L. Field, Dynamic federation of grid and cloud storage, Phys. Part. Nuclei Lett. (2016) 13: 629. https://doi.org/10.1134/S1547477116050186

[24] F. Berghaus *et. al.*, Federating distributed storage for clouds in ATLAS, J. Phys.: Conf. Ser. 1085 032027 (2018). doi:10.1088/1742-6596/1085/3/032027

[25] M. Ebert *et. al.*, Using a dynamic data federation for running Belle II simulation applications in a distributed cloud environment, EPJ Web of Conferences 214, 04026 (2019). https://doi.org/10.1051/epjconf/201921404026

[26] F. Berghaus *et. al.*, The Dynafed data federator as grid site storage element, http://heprcdocs.phys.uvic.ca/presentations/chep-berghaus-dynafed-2019.pdf, CHEP 2019, Adelaide, Australia, proceedings in preparation.