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. A decade after its initial design and implementation, cloudscheduler has been modernized to take advantage of new software designs, improved operating system capabilities and support packages. The result is more resilient and scalable system, with expanded capabilities.

1 Introduction

We describe a 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 users. 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 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. 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.

We remark that there are other strategies and software for utilizing clouds in particle physics, such as Vcycle [5], VMDIRAC [6], and HTCondor [7]. The design of these systems is different and complementary to our model. For further information on the other solutions, we refer the interested reader to the cited papers.

2 Overview

The architecture of the distributed compute cloud has been described in a number of papers [8, 9]. 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 [10], making it an ideal fit for a dynamic cloud environment where resources (VM instances) appear and disappear.

Cloudscheduler is a framework of components written in Python 3 and uses a MariaDB database [11] to track the state of the clouds and the HTCondor pool (see Fig. 1). 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.

Cloudscheduler has *pollers* that fill the MariaDB with information from the clouds and HTCondor. New entries are added to 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 example, completed jobs/VMs) is removed. 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.

Although Fig. 1 shows only one HTCondor instance, cloudscheduler 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 an *HTC Agent* to provide the correct security context when issuing requests to the HTCondor client on the VM.

3 Workflow

The VM Scheduler retrieves virtual tables (views) from the MariaDB to determine its actions. A view relates data from multiple tables, perform calculations, sort, group, select, concatenate, resort and regroup to derive the information required. The primary views are the job queue and the compute resources.

We briefly describe the steps to create the job queue view. Jobs are grouped according to their resource requirements such as cores, disk, and memory (called a "job-group"). For example, ATLAS has jobs requiring either 1 or 8 cores.

The next step is to find the VM flavours on each cloud that can satisfy the job requirements of the job-group. The result is a list of (Cloud name:VM flavour) pairs. If there are multiple entries for a single cloud, then the the VM flavour with the fewest cores and/or the least amount of memory is selected. If a cloud has no VM flavour that meets the requirements,

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

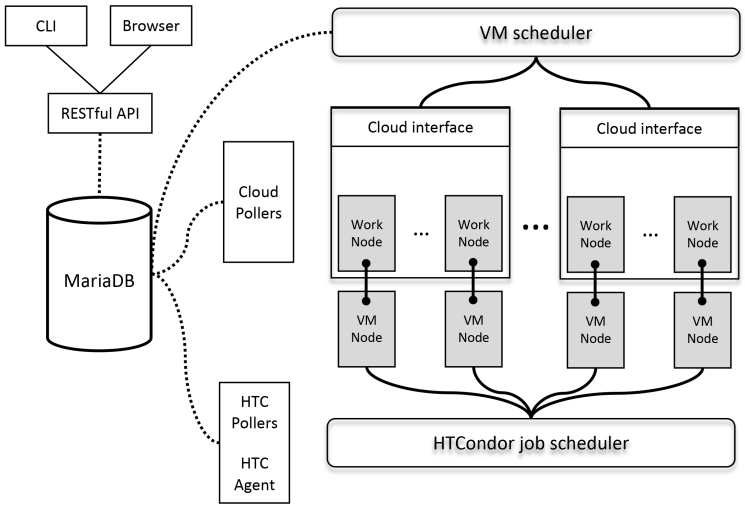

Figure 1. Overview of the cloudscheduler and HTCondor distributed compute cloud. The primary change is the introduction of the MariaDB for keeping track of the state of the system and "pollers" to gather information from the clouds and the job scheduler. The information in the database is used for scheduling,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

then that cloud is not included in the list of VM flavors (and will not be selected to boot VMs for this job-group).

The result of the job queue view is the table shown below, listing the group name, the number of idle jobs, and a list of (Cloud name:VM flavour) for each unique job type (only a subset of the columns in the table is shown). Each row corresponds to a unique job type. The first row is for ATLAS 8-core jobs, the second row is for Belle II production (low I/O) jobs and the third row is for Belle II analysis (high I/O) jobs. The list of (Cloud name:VM flavour) pairs is ordered by the priority of the cloud (called the cloud-priority). The cloud-priority can, for example, give lower priority to commercial clouds. Clouds that have the same priority are ordered alphabetically based on their cloud name.

Group	Idle jobs	Cloud name : VM flavour
atlas	86	arbutus-nf:c16-60gb-392, arbutus:c8-30gb-186, cc-east:c8-30gb-430, chameleon:m1.xxlarge, otter:b8
belle	1	arbutus:c8-30gb-186, cc-east:c4-15gb-205, desy:m1.xlarge, ecdf-b:m1.xlarge, otter:b8
belle	91	arbutus:c8-30gb-186, otter:b8

The next step is to to determine how many VMs can be started for each (Cloud name:VM flavour) pair. The following table shows the result of the compute resources view that returns the group name, the (Cloud name:VM flavour) pair, and the number of Available VM slots that can be started on this cloud. In this example, the arbutus cloud has the capacity to start 109 VMs with the c8-30gb-186 flavour for ATLAS and 184 VMs with the same flavour for Belle II. The table was selected to show only the results of a query for a VM flavour of arbutus:c8-30gb-186, which is the optimal VM flavour for both projects (an unrestricted query returns over 100 rows).

Group	Cloud name : VM flavour	Available VM slots
atlas	arbutus:c8-30gb-186	109
belle	arbutus:c8-30gb-186	184

The compute resources view provides the *VM scheduler* with the information it requires to boot new VMs. Prior to issuing VM boot requests, a number of conditions are tested to avoid starting too many VMs. For example, if there are more than five VMs in a *starting* state on a cloud, then no VMs will be started on that cloud. In addition, if there are more that ten *idle* VMs (i.e. not running any jobs), then no VMs are started for that job type group.

If the above criteria are satisfied and the list of (Cloud name:VM flavour) pairs in the compute-view table shows available VM slots (see above table), then the *VM scheduler* will issue up to five VM boot requests on each cloud in the group. At the end of the boot process, the VMs are contextualized with the configuration required by cloudscheduler (e.g. HTCondor), the particle physics software (e.g. CVMFS file system) and the experiment software.

If a VM remains idle for an extended period of time after completing one or more jobs, then either the job queue is empty, there are no more queued jobs that can run in the VM, or the VM developed problems communicating with the HTCondor server. In this situation, the *VM scheduler* sets the retire-flag in the database entry of the VM. If the retire-flag of the VM in the MariaDB is set, then the *HTC machine poller* sends retire messages to the *HTC Agent*. On receiving a retire message, the *HTC Agent* issues a "condor_off" command to both the HTCondor startd and master daemons on the VM instance, causing all the partitions to be marked "retiring" and preventing them from accepting new jobs. However, jobs currently running on the VM are allowed to complete. Once the jobs on the VM are finished, then the *HTC machine poller* sends a request (via the cloud API) to the cloud to shut down that VM instance.

4 Configuration

Users can be assigned different access privileges and can be members of multiple groups. Groups are defined, in our case, to be particle physics experiments or a development and testing area[2]. Each group adds its compute clouds to its configuration, and users in the group have the ability to add and manage the clouds.

The VM images and SSH keys[3] can be uploaded by logging in directly to the cloud (e.g. via the Openstack or Amazon dashboards) or via the Cloudscheduler GUI. One can define a default VM image and default SSH keys that will be automatically distributed to all Openstack clouds. Cloudscheduler keeps track of the VM images in the MariaDB. As the list of VM images can change, the *cloud poller* periodically queries the clouds for the list of available VM images as well as VM flavours and cloud quotas to keep the MariaDB up to date.

5 VM contextualization for particle physics applications

At the end of the boot process, the VMs are contextualized with the required software and the VM is configured for the application software [12]. The contextualization is controlled by metadata that is passed to the "cloud-init" process [13].

[2]Openstack has projects and users. Previously, Openstack used tenants before changing to projects. As cloudscheduler uses other types of clouds, it was decided to identify ensembles of users as groups to distinguish it from Openstack projects.

[3]The SSH keys allow the user to log into a VM instance for debugging purposes.

The contextualization of the VM instances for the ATLAS and Belle II experiments is very similar. Both use the micro-CernVM virtual machine image [14], the CernVM File System (CVMFS) [15], and optionally, *Shoal*, a dynamic web cache locator service [16].

The small 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 files from CVMFS [17]. The CVMFS client is configured to use a specific squid cache; however, the CVMFS client can be configured 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 CPU benchmark suite is run on the VM. Both the HTCondor client and the CPU benchmark are retrieved from CVMFS. The necessary host certificates and SSH keys are added to the VM.

It is important to keep track of the resources delivered by a cloud to an experiment. We configure the VMs to write out their status every 15 minutes to Elasticsearch. The status includes configuration information, the fast benchmark and the timing information (obtained from /proc/stat). We are currently working on reporting our accounting numbers back into APEL for ATLAS (the accounting information for Belle II will be added once the reporting system is operational) [18].

6 Operations and status

The distributed compute cloud has been running ATLAS and Belle II jobs for many years. ATLAS and Belle II are 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 [19]. 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 [20]. 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 total workload (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 [21, 22]. 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 [23–25].

7 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.

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] 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

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

[8] 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

[9] 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

[10] 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

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

[12] K. Keahey and T. Freeman, Contextualization: Providing One-Click Virtual Clusters, IEEE Xplore: 06 January 2009, DOI: 10.1109/eScience.2008.82

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

[14] 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

[15] 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

[16] 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

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

[18] APEL is an accounting tool that collects accounting data from sites participating in the EGI and WLCG infrastructures, https://wiki.egi.eu/wiki/APEL

[19] 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

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

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

[22] 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

[23] 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

[24] 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

[25] 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.