Managing remote cloud resources for multiple HEP VOs with cloudscheduler

Colson Driemel[1], *Marcus* Ebert[1,*], *Randall* Sobie[1], and *Tristan* Sullivan[1]

[1]High Energy Physics Research Computing Group, University of Victoria, BC, Canada

Abstract.
Cloudscheduler is a system to manage resources of local and remote compute clouds and makes those resources available to HTCondor pools. It examines the resource needs of idle jobs, then starts virtual machines (VMs), sized accordingly, on allowed clouds with available resources. Using yaml files, cloudscheduler then provisions the VMs during the boot process with all necessary tools needed to register with HTCondor and run the experiment's jobs. Although we have run cloudscheduler in its first version for ATLAS and Belle-II workloads successfully for more than 10 years, we developed cloudscheduler version 2 (CSV2), a complete overhaul and modernization of cloudscheduler. The new system is used successfully in production for Belle-II, ATLAS, DUNE, and *BABAR*. In addition to using cloudscheduler version 2 as a WLCG site, we also run it as a service for other WLCG sites, and the Canadian Advanced Network for Astronomical Research (CANFAR) group uses its own instance of CSV2 for their astronomy workloads. In this paper, we report on our experience in operating CSV2 for different experiment's jobs, running on up to 10,000 cores across all experiments and clouds in North America, Australia, and Europe. We will also report on how to correctly account for the resource usage in the WLCG APEL system, how the monitoring works, as well as on the integration of different clouds and how to use resources opportunistically.

1 Introduction

Cloudscheduler version 1 was written more than a decade ago to make use of cloud resources as batch system worker nodes [1]. While it was successfully run for ATLAS and Belle-II workloads by the High Energy Physics Computing Research (HEP-RC) group at the University of Victoria (UVic)[2], it was written in Python 2, which reached end of life in 2020. Rewriting the system in Python 3, the opportunity was used to design a new system from the ground up: cloudscheduler version 2 (CSV2). The technical details about CSV2 have been published in [3] and an overview of the workflow is shown in Fig.1. Access to CSV2 and resources in CSV2 are organized by groups. CSV2 is connected to a single HTCondor system [4] for each group, which can run on the CSV2 machine or anywhere else, as well as has clouds defined in each group. Depending on the resource needs of the jobs in a HTCondor queue, CSV2 can start Virtual Machines (VMs) with the appropriate resources on a cloud. Once no more jobs are in the queue that can use the available resources in started VMs, those

*e-mail: mebert@uvic.ca

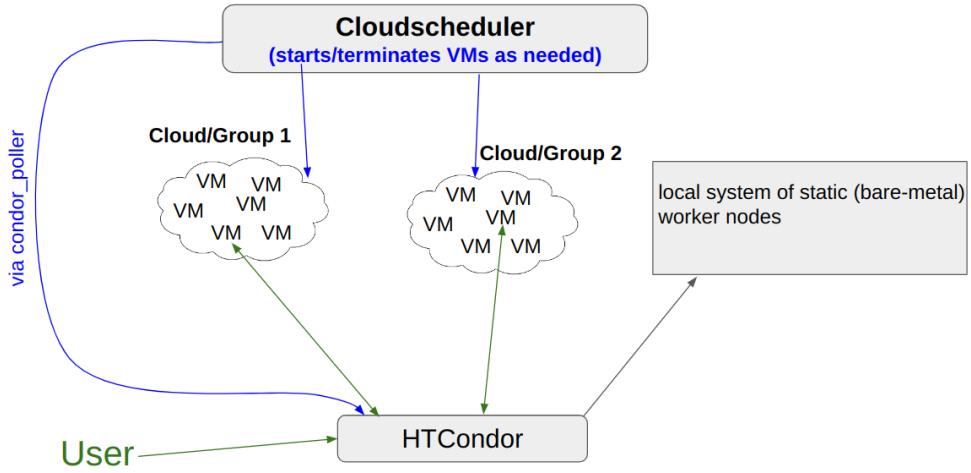

Figure 1. General interaction between users submitting jobs, the HTCondor batch system and CSV2 interacting with clouds for different groups.

VMs get retired in HTCondor. CSV2 terminates the VMs once they are no longer registered with HTCondor.

Jobs can be submitted to HTCondor manually or through an experiment's job management systems when used as a grid compute element for the World Wide LHC Computing Grid (WLCG). A single instance of CSV2 can manage jobs in multiple HTCondor systems and job resources for multiple experiments and CSV2 groups in parallel. The customization of VMs can be done via cloud-init, in which case the contents of group specific yaml files are transferred to the cloud and to the VM on startup. Those yaml files are managed within CSV2's web interface and can be edited in the browser. It is also possible to specify the order in which those yaml files get executed. This allows the use of a small, general purpose, VM image which is then customized at boot time for any given experiment via the yaml files. We use CSV2 to run jobs for ATLAS[5], Belle-II[6], and DUNE[7] in this way, using a CERNVM image[8] as lightweight general image.

If an experiment needs a very specialized image, that can also be used without any customization applied. We use a fixed image provided by *BABAR*[9] for their jobs. The *BABAR* software environment was frozen more than a decade ago, and only this frozen environment is validated for physics analyses [10].

2 CSV2 as a WLCG compute site

In this section we will detail how to operate CSV2 as a WLCG site using our operations for ATLAS and Belle-II as an example. We will show what is needed to successfully run jobs for the different experiments, as well as how to use the same resources for multiple experiments in a way that one experiment's jobs get priority use of cloud resources. At the end of this section we will show how to report job statistics to the EGI accounting portal[11].

We assume that for each experiment an HTCondor server is available and that the experiments' computing systems are already configured to send jobs to a specific HTCondor system, for example ATLAS submits via Harvester[12] and Belle-II via a DIRAC site director[13].

2.1 Setup of CSV2 for multiple experiments

To make use of CSV2, one CSV2 group per experiment needs to be created via the web interface, specifying the hostname of the HTCondor server. After configuring clouds to be used by the group, some defaults, like the VM image to be used, can also be defined. After a group is created, CSV2's condor poller should be installed on the HTCondor server used by this group. Instructions on how to do that can be found in the web interface of CSV2 under the menu item "HTCondor plugin". If successful, the status page for the group will display information from the HTCondor server, like the current number of jobs, as well as if the condor poller is installed correctly.

To be able to execute jobs on VMs, cloud resources must be configured in the "Cloud Config" menu. Currently, Openstack and Amazon clouds are supported. Once a cloud is configured, available resources, images, flavors, and other cloud specific options can be selected. If an image is specified in the group configuration as default and it is not on a newly configured cloud yet, then it will automatically be transferred to each cloud that this CSV2 group has access to. It is also possible to upload images to clouds via the CSV2 web interface.

The above process can be repeated for as many CSV2 groups as needed. For a single WLCG site, a new CSV2 group should be created for each experiment for which jobs are to be run.

2.2 Opportunistic usage of resources between groups

For our WLCG site, two different CSV2 groups are used in CSV2, one for ATLAS (atlas-uvic) and one for Belle-II (belle). While both groups have different clouds configured, clouds common to both are also used. In this case it is useful that one of the two groups gets priority access to the resources, otherwise both would compete for the same resources. This is possible by utilizing the "Cores Softmax" in CSV2's cloud config menu. The "Cores Softmax" is a property that specifies how many cores can be used in total by all VMs on a specific cloud, no matter to which CSV2 group they belong. In the case of ATLAS and Belle-II, both have access to a cloud project with a total of 4,000 cores available. We give priority to Belle-II and set the "Cores Softmax" for Belle-II to 4,000 cores for this cloud. ATLAS jobs on the other hand should only use this cloud in an opportunistic way, when there are not enough Belle-II jobs available to use all cloud resources. This can be accomplished by configuring the "Cores Softmax" value for the ATLAS group lower than for the Belle-II group. We found that setting it to 3,000 cores is a good choice for us.

What happens then is that when Belle-II has enough jobs, all available resources are used by Belle-II jobs and none is used for ATLAS jobs. When Belle-II does not have enough jobs to use the active resources, CSV2 will automatically terminate idle VMs. Once the core count of used VMs on this cloud falls below 3,000, CSV2 sees that there are less cores used than specified for ATLAS in the "Cores Softmax". It will start VMs to run ATLAS jobs since less than 3,000 cores are used on this cloud in total. If at some point no more Belle-II jobs are available, then CSV2 will start VMs for ATLAS until 3,000 cores are used on this cloud.

When at a later point new Belle-II jobs are available, then CSV2 can still start VMs for Belle-II since the Belle-II "Cores Softmax" specifies 4,000 cores for Belle-II. Once new Belle-II VMs are started, more than 3,000 cores are used in active VMs on this cloud and CSV2 will automatically start to send the retire command to the HTCondor services on some of the ATLAS VMs. When those ATLAS VMs are no longer registered with HTCondor, they will be terminated by CSV2, thus trying to keep the total used cores on this cloud to less than or equal to 3,000 for the ATLAS CSV2 group. This frees up resource to be used by Belle-II jobs on this cloud. Over time, this process increases the used cores for Belle-II and reduces

the used cores for ATLAS, as long as Belle-II has enough jobs in the queue. If not, then CSV2 starts again shifting resources over to ATLAS.

By using the "Cores Softmax" in that way - specifying different limits for different groups - priority use of cloud resources shared between different experiments can be achieved. When there is a large difference in the "Cores Softmax" value, then there are always enough resources available for jobs of the experiment that has the larger value. But there may be times when a larger number of cores is not used at all. On the other hand, if there is only a small difference in the "Cores Softmax" value, then nearly all resources are always used, but it may take longer until jobs for the experiment that has the large value will scale up. The turn-over between the different experiments depends mainly on the average job run time and how long it takes until a retired VM is deregistered from HTCondor.

2.3 Job accounting

To have the resources used by jobs of a particular experiment accounted for, WLCG sites usually use the APEL service[14] to report job statistics to the global EGI accounting service. Using CSV2 and HTCondor as WLCG site, this is also possible. To do so, we use the HTCondor job statistics. While nearly all needed values are stored in a job's ClassAd, the benchmark value of the machine used is not. This value is needed to scale values from different machines to account for the different performance of CPUs.

To add the missing values to the HTCondor job ClassAd, first we needed to know the benchmark value for the different machines that are used on the clouds, either by asking the cloud admins or by running a specific benchmark on different VMs. Once the numbers are known, they can be transferred to a VM at startup via the yaml files mentioned before. In addition, a script is transferred to the VM at startup, which determines the correct benchmark value for the machine. For other accounting purposes using different benchmarks, benchmarking can also be done when a VM is started and its result used later on.

Once the benchmark value is known for a specific VM, it needs to become known to HTCondor. This is done by creating a file `/etc/condor/config.d/benchmark` containing the benchmark value and a command to add it to the `STARTD_ATTRS`. For example, using the HEPScore benchmark[15] that gave a value of 30.3, the content of that file would be

```
HEPScore = "30.3"
STARTD_ATTRS = $(STARTD_ATTRS) HEPScore
```

In addition, the main HTCondor server needs to have the options `WantIOProxy = True` and `SUBMIT_ATTRS = $(SUBMIT_ATTRS) WantIOProxy` in its configuration.

WLCG jobs are usually executed on worker nodes by using a job wrapper script, which prepares the environment and then starts the real job. In this wrapper script on the worker node VM, the benchmark value, which is now known to the worker, needs to become part of the job ClassAd using `condor_chirp`. For the above example, the following needs to be added to the job wrapper script:

```
#add benchmark value to job ClassAd
 hepscore=$(condor_config_val HEPScore)
 condor_chirp set_job_attr HEPScore $hepscore
```

All of the script customization on the worker node VM are done via CSV2 using yaml files as described before.

With the above changes, all values needed for the job accounting are available via `condor_history`, which displays information about finished jobs. A shell script is used

to parse the output of `condor_history` and to prepare a text file in the format needed by the APEL accounting scripts. A cron job then reports the values to the EGI accounting using the prepared text files.

3 CSV2 as a compute service for other WLCG sites

Since CSV2 can manage multiple groups independently, other WLCG sites that have clouds available can make use of our HEP-RC CSV2 instance[16]. There are different ways to accomplish this and it depends on whether the site admins want to manage the compute themselves or just provide a cloud and let the admins of CSV2 handle the rest.

3.1 Cloud compute of WLCG sites managed by the CSV2 admins

In this case, no access to CSV2 by members of other WLCG sites is needed, and a single CSV2 group for an experiment can be created as was described in section 2.1. Clouds of all WLCG sites that are to run jobs for the same experiment should be added in the cloud config of this group.

To distinguish between clouds and to have jobs for a specific site run only on their own cloud, target aliases need to be set in the configuration and specific clouds added to each alias. Jobs coming in then need to have in their requirements an additional string specifying an alias. One of the sites we run ATLAS jobs for as a service is the Edinburgh GridPP site (ECDF)[17]. The alias we have configured for this site has the same name as the corresponding ATLAS job queue and only the ECDF cloud is configured for jobs that require this target alias. In this case, the addition to the requirements string is `(group_name=?="atlas-cern"&&target_alias=?="uki-scotgrid-ecdf_cloud")`. The `group_name` is the name of the CSV2 group used for those jobs and lets CSV2 identify in which group those jobs should run. Using target aliases, it is possible to have jobs for multiple WLCG sites run through a single CSV2 group.

3.2 Cloud compute of WLCG sites managed by the sites

When a WLCG site wants to manage their own resources with CSV2, they can install their own instance or use an existing one. On our CSV2 instance, we let site admins from Melbourne, Australia, and from DESY, Germany, manage their own resources. To do so, we have created a group for each site as described before. Since CSV2 has a built-in authorization interface, we can add multiple users to each group which gives site admins access to their group in CSV2. Authorization is preferably done using grid certificates. Once the site admins have access to their group, they put in their cloud information as described before and also all other resource information that are needed to run jobs.

4 Opportunistic cloud usage

Different than in section 2.2, there is also the possibility that a single cloud has unused resources. Cloud admins may not want to have their resources idle and look for users that can make use of idle resources in an opportunistic way. We tested this approach with some of our clouds and found it to work well with CSV2.

To make use of opportunistic resources, first the cloud admins needed to increase the resources that are available to our cloud project. For example, if the cloud project can utilize

under normal circumstances 1,000 cores, but another 1,000 cores could be used in an opportunistic way, then the cloud admins should increase the usable cores to 2,000 for the project. To not always use 2,000 cores however, the previously introduced "Cores Softmax" property can be used to limit normal usage of the cloud CPU resources. Setting the "Cores Softmax" to 1,000 cores would limit the usage to the regular allocation even if up tp 2,000 cores could be used. To make this process dynamic, the cloud admins introduced a new property for our cloud project called "dynamic-cores" and set it to an value of their liking to have the cloud resources used as much as possible. Using the CSV2 command line interface, we can set the "Cores Softmax" to the "dynamic-cores" value obtained from the cloud project using a cron job. That way, CSV2 would not allow more cores to be utilized in the cloud project than specified in the "dynamic-cores" which is controlled by the cloud admins.

When there are idle resources on the cloud, the cloud admins can increase the "dynamic-cores" value for our project. This way, CSV2 will automatically use more resources when available. When those resources are needed by other groups again, the cloud admins can simply decrease our "dynamic-cores" value and CSV2 will automatically retire and terminate VMs until the new target specified in the new "dynamic-cores" value is reached.

5 Monitoring

One important aspect of running a compute site is the monitoring of the resources, both for accounting purposes and to easily find operational issues. CSV2 has a status page displaying all information about the clouds and their usage, jobs that are currently idle or running, and general server status information. Issues that are found are clearly marked and it is also possible to send emails about found issues to the CSV2 administrator. All values displayed on the status page can be plotted for a chosen time range. This can be used for accounting purposes, as well as to spot issues with job execution or VM startup on a cloud.

Figure 2 shows the status display for the CSV2 ATLAS group where we run as a service for other sites, as well as the used cores over the last year as an example of the timeline for values displayed. The status page also shows the target aliases used to send jobs to specific clouds only, information about how many cores jobs in the queue need, and certificate information when used by HTCondor.

Limited information is also available on a public status page [16]. Figure 3 shows the job information in the different queues, and the cloud based information are shown in Fig.4, both part of the public status page.

6 Summary

We successfully run CSV2 as a WLCG compute site and also as a service for other WLCG sites. A single CSV2 instance manages resources for jobs sent by Belle-II, ATLAS, DUNE, and *BABAR* on clouds in North America, Europe, and Australia. CSV2's group management is used to separate workloads of different WLCG sites and experiments. Administrators of other WLCG sites are able to log in to the web interface of our CSV2 instance and manage their own cloud resources as they see fit, independently of everyone else. We have shown how the EGI job accounting can be accomplished with CSV2 utilizing the HTCondor job history. We have also shown how to use shared cloud resources for different groups, where one group can use those resources in an opportunistic way. We also tested a way to opportunistically use idle cloud resources, usually allocated to cloud projects independent of CSV2, where the cloud administrators are able to control the amount of opportunistic usage. The built-in monitoring and time line plotting capabilities of CSV2 were very useful in getting an overview of the

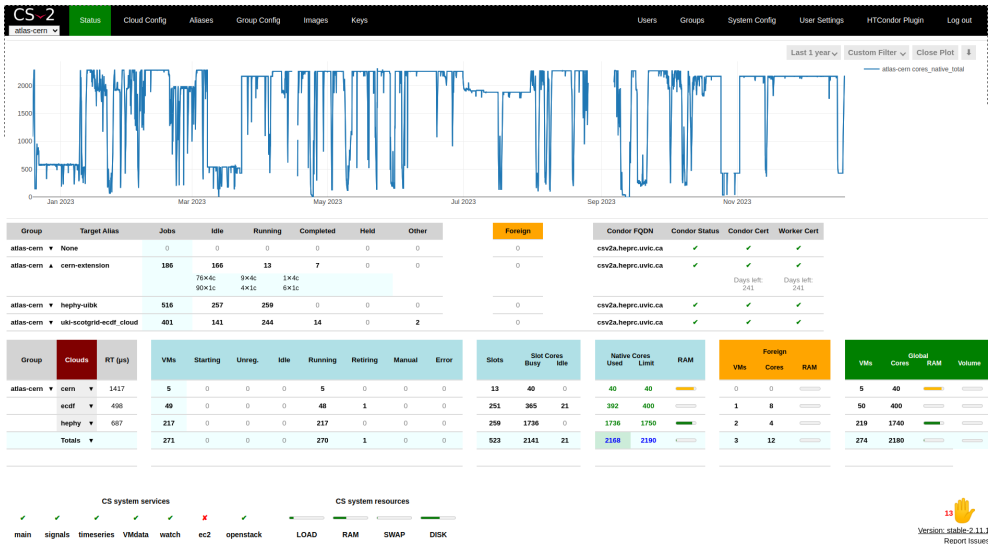

Figure 2. Internal CSV2 status page for the ATLAS group where we run jobs as a service for other WLCG sites. The jobs status for each site, the utilized clouds, and the timeline for the total number of used cores over the last year is shown.

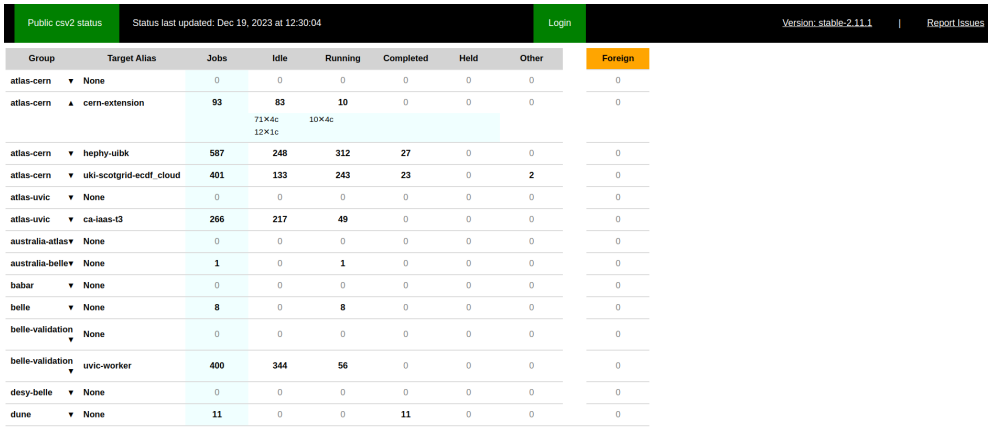

Figure 3. Our CSV2 public status page with job information for all experiments and groups.

utilization of cloud resources as well as to easily find issues in operation. We have shown that using CSV2 with distributed cloud computing resources provides all the functionality of a traditional WLCG computing site, including stable operation, accounting, and monitoring, while also providing significant flexibility in how the resources are used.

References

[1] A. Charbonneau et al., Journal of Physics: Conference Series **341**, 012003 (2012),

[2] *The High Energy Physics Research Computing group at the University of Victoria, Canada*, accessed: Sept. 2023, `http://heprc.phys.uvic.ca/`

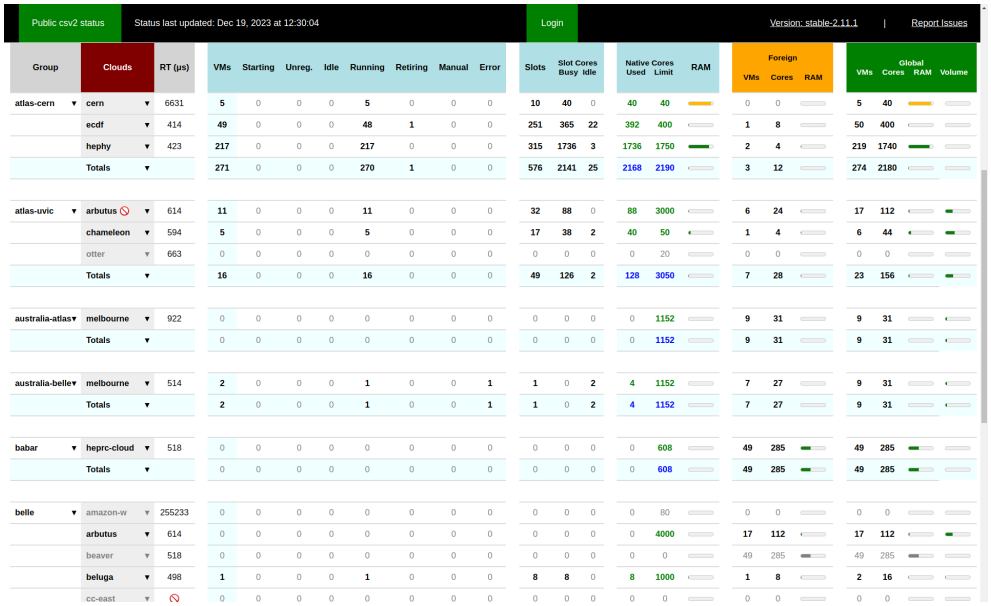

Figure 4. Part of our CSV2 public status page with cloud-based information for all experiments and groups.

[3] F.Berghaus et al., Computing and Software for Big Science **4,4** (2020),

[4] *HCondor Software Suite*, accessed: Sept. 2023, `https://htcondor.org/`

[5] *The ATLAS experiment*, accessed: Sept. 2023, `https://atlas.cern/`

[6] *The Belle-II experiment*, accessed: Sept. 2023, `https://www.belle2.org/`

[7] *The DUNE experiment*, accessed: Sept. 2023, `https://www.dunescience.org/`

[8] *CERNVM APPLIANCE*, accessed: Sept. 2023, `https://cernvm.cern.ch/appliance/`

[9] *BABAR at the SLAC National Accelerator Laboratory*, accessed: Sept. 2023, `https://www-public.slac.stanford.edu/babar/`

[10] *DPHEP presentation of the BABAR LTDA computing system at SLAC*, accessed: Sept. 2023, `https://indico.cern.ch/event/209688/contributions/1501412/`

[11] *EGI Accounting Portal*, accessed: Sept. 2023, `https://accounting.egi.eu/`

[12] Barreiro Megino et al., EPJ Web Conf. **245**, 03010 (2020)

[13] *Dirac, the interware*, accessed: Sept. 2023, `https://dirac.readthedocs.io/`

[14] *APEL accounting tool*, accessed: Sept. 2023, `https://apel.github.io/`

[15] *The HEPScore application*, accessed: Sept. 2023, `https://gitlab.cern.ch/hep-benchmarks/hep-score`

[16] *The hep-rc cloudscheduler instance*, accessed: Sept. 2023, `https://csv2.heprc.uvic.ca/public/`

[17] *The Edinburgh GridPP site*, accessed: Sept. 2023, `https://www.ph.ed.ac.uk/particle-physics-experiment/our-activities/gridpp`