EPJ Web of Conferences **251**, 02059 (2021) https://doi.org/10.1051/epjconf/202125102059
CHEP 2021

Opportunistic transparent extension of a WLCG Tier 2 center using HPC resources

R. Florian von Cube[1,*], *René* Caspart[1], *Max* Fischer[1], *Manuel* Giffels[1], *Eileen* Kuehn[1], *Günter* Quast[1], and *Matthias J.* Schnepf[1]

[1]Karlsruhe Institute of Technology

Abstract. Computing resource needs are expected to increase drastically in the future. The HEP experiments ATLAS and CMS foresee an increase of a factor of 5-10 in the volume of recorded data in the upcoming years. The current infrastructure, namely the WLCG, is not sufficient to meet the demands in terms of computing and storage resources.
The usage of non HEP specific resources is one way to reduce this shortage. However, using them comes at a cost: First, with multiple of such resources at hand, it gets more and more difficult for the single user, as each resource normally requires its own authentication and has its own way of accessing it. Second, as they are not specifically designed for HEP workflows, they might lack dedicated software or other necessary services.
Allocating the resources at the different providers can be done by COBaLD/TARDIS, developed at KIT. The resource manager integrates resources on demand into one overlay batch system, providing the user with a single point of entry. The software and services, needed for the communities workflows, are transparently served through containers.
With this, an HPC cluster at RWTH Aachen University is dynamically and transparently integrated into a Tier 2 WLCG resource, virtually doubling its computing capacities.

1 Usage of non-community specific resources

Several HEP experiments are provided with computing resources through the *Worldwide LHC Computing Grid* (WLCG). It consists of a network of hundreds of computing centers, organized in a tiered structure. At Tier 0 is the "CERN Data Center" in Geneva. It accounts for performing the first reconstruction of the experiments raw data and storing both the raw data and initial reconstruction. The 13 Tier 1 centers, all around the globe in participating research facilities, mirror parts of the raw and reconstructed data, each, such that every file exists at least twice. They also perform further reprocessing of the data and provide the experiments with compute resources for data analysis and event MC (Monte Carlo) simulation. The roughly 160 Tier 2 centers, located at universities and other research facilities, satisfy the need for computing power and storage for user analysis and also MC simulation.

Many research groups have access to additional compute resources, commonly called Tier 3 resources, which not necessarily are accessible through grid compute elements. Those

*e-mail: ralf.florian.von.cube@cern.ch

resources are not officially part of the WLCG and operated by the local groups, which take care of providing the necessary environments by themselves. However, Tier 3 resources play an important role for the groups in providing sufficient computing resources for the analyses, as the official resources sometimes do not meet the demand.

The group might also have access to third-party resources like computing clusters operated by their university, shared science clouds, or e.g. resources at commercial cloud providers. The advantage of those resources is, that the group does not have to do the administration, however as they are often not specifically designed for the community, lacking necessary software or services have to be provided through different means.

Both the ATLAS and the CMS experiments at the LHC expect to record 5-10 times more data in the HL-LHC era by 2026 than they do today. The WLCG resources as of today will not be sufficient to provide the necessary amount of storage and CPU for data processing, even with the expected flat funding and the increase due to technological advances. [1] In order to tackle this shortage, the usage of existing additional Tier 3 resources should be facilitated and other possible resource providers should be made available.

Each provider requires its own authentication and has a different way of accessing the resources, e.g. in terms of the batch system used. The variety of the technologies implies increasing difficulty in managing the resources from the single user point of view. By integrating all the resources into a common *overlay batch system* (OBS), their usage can be made transparent for the users. The user authenticates against and submits the jobs to the OBS, which then dispatches them to the most fitting resource. In this way, the user can benefit from a whole plethora of resources without struggling with different authentication and access methods.

However, having computing time at different providers, it remains the question, where to allocate which type of resource. This question is addressed by the OBS which can decide where to dispatch the jobs. The resource management is then provided by COBALD/TARDIS as described in the following.

2 Resource integration with COBALD and TARDIS

In order to automate the integration of external resources, KIT develops "COBalD – the Opportunistic Balancing Daemon" [2] (COBALD) and the "Transparent Adaptive Resource Dynamic Integration System" [3] (TARDIS). The integration is performed through drones, which are a generalization of the pilot concept, also used e.g. in the WLCG [4]. A drone represents a worker-node of the OBS and is responsible for setting up the required software to run jobs from the OBS and integrating itself into the OBS. The steering of the drones, thus the allocated resources, is done by COBALD and TARDIS.

To quantify how well a resource fits to the current job mix, a set of indicators $\mathcal{I}$ is taken into account and the metrics

$$\text{allocation} = \max_{i \in \mathcal{I}} \left(\frac{utilized(i)}{allocated(i)} \right), \text{ and} \tag{1}$$

$$\text{utilization} = \min_{i \in \mathcal{I}} \left(\frac{utilized(i)}{allocated(i)} \right) \tag{2}$$

are defined. The set $\mathcal{I}$ can contain quantities such as CPU cores, or memory. But also other indicators, as e.g. software license are possible. With those definitions, allocation is a measure, how "full" a resource is, i.e. whether more jobs fit on the resource, while utilization represents the suitability of the resource for the jobs. A low utilization might suggest that the resource type does not fit to the jobs, high utilization in turn, hints a good fit.

COBaLD provides an abstraction layer to resource pools and abstract controllers for deciding whether to increase or decrease the number of a specific resource. For this it combines alike drones on the same resource to one pool and handles the quantities demand and supply of each pool. The latter measures the number of drones of a specific resource currently available in the OBS, the former is adjusted depending on the metrics defined above (allocation and utilization). The higher the value of allocation, the more resources are needed by the OBS, thus COBaLD increases the demand in this pool. If the value of utilization in turn is small, the resource is not the optimal fit for the current jobs, thus the demand should be decreased in this pool.

With this design, each type of resource constitutes one pool in COBaLD with the quantities demand and supply. This allows for horizontal scaling, as the decision how many drones to integrate of each resource type is taken on a per-resource basis. By reacting on the decisions of the OBS, rather than trying to predict the future resource need, the system always chooses the currently best fitting resource. [5]

The metrics allocation and utilization are determined by TARDIS which uses COBaLD as a feedback loop to decide, on which resource to increase or to decrease the number of drones. TARDIS can interface with several types of resources, i.e. the batch systems HTCondor, MOAB and SLURM, and the APIs CloudStack and OpenStack in order to determine the metrics and perform the adjustments of number of drones, as indicated by COBaLD. [6] The specific type and configuration of the drone depends on the resource. However the drones start up the required software for an OBS worker node and integrate themselves into the OBS. After this initialization, the drones are expected to be ready to accept jobs from the OBS. As of today, TARDIS supports HTCondor and SLURM as OBS. The drones, in case of a batch system, might consist of a bash script, which TARDIS submits to the batch system. This script then sets up the necessary environment and starts up the OBS worker node component, required to run jobs. Another type of drone might be a container-image, which provides the necessary services after startup. In case of the cloud computing platforms, a virtual machine and the necessary services are started.

Container and virtualization

Depending on the design of the drone, which is determined by the resource it is running on, there still might be e.g. some software missing to run the actual payload from the OBS. However, in most cases, this can be resolved using containerization technologies like Docker [7], or Singularity [8, 9]. By provisioning container-images, which set up the required environment, those services can be supplied. Also file systems like CVMFS [10] repositories can be made available inside the container via bind mounts. For the jobs started in the container by the OBS, the realized environment is the same as on any other community-resource.

3 Setup in Aachen

The physics department at RWTH Aachen University operates a Tier 2 computing center as part of the *Worldwide LHC Computing Grid* (WLCG). The cluster consists of about 7200 cores. While the pledge to the WLCG is around 5100 cores, the additional cores are a computing resource for multiple local research groups.

For research and teaching, the university operates the high-performance computing resource “Cluster Aix-la-Chapelle” (CLAIX). The cluster is a classical HPC system, that has to provide resources to very different research groups and is therefore not dedicated to physics. It consists of roughly 75.000 cores in total and high performance parallel file systems.

As proof-of-concept of extending a WLCG Tier 2 resource with an HPC cluster, CLAIX resources were integrated into the Aachen Tier 2 in a transparent manner for end users and the CMS experiment. For this, an instance of COBALD/TARDIS is running on a service machine at the Tier 2. TARDIS submits batch jobs to CLAIX' local resource management system (LRMS), the SLURM batch system. The batch job consists of a bash script, setting up a temporary HTCONDOR installation using pre-compiled binaries with the permissions of an unprivileged user. The HTCONDOR-daemon which is started up subsequently, connects to the central manager of the Tier 2 and is then able to accept regular WLCG jobs. Any job matched to the CLAIX resources, is started by HTCONDOR in its own SINGULARITY container which provides the WLCG environment, expected to be provisioned at official WLCG resources. That is, the container image supplies software in specific versions and the necessary CVMFS repositories mounted on the host are bind-mounted into the started container.

If the matched job is a pilot job from the CMS experiment, it in turn configures and starts up an HTCONDOR-daemon, which connects to the experiments central manager. Then jobs from the experiments batch system are matched to the resource. The final payloads are run in additional SINGULARITY containers, provided and taken care of by the experiment. The process tree of running such a payload on the CLAIX resource is depicted in figure 1.

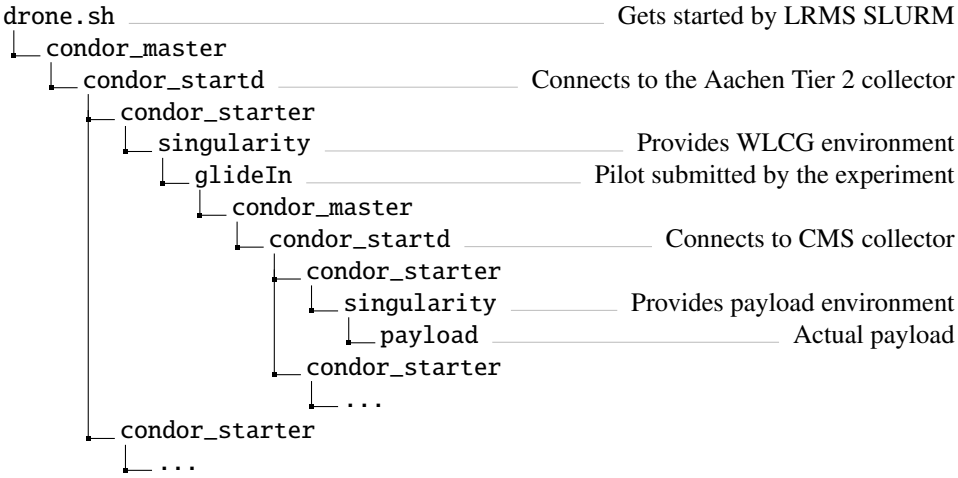

Figure 1. The process tree on the CLAIX worker node and the function of each process. Technically, each drone can run multiple glideIn.

Challenges

As the Aachen Tier 2 is protected by a firewall, a *Condor Connection Broker* (CCB) is used for communication of the daemons running on CLAIX with the central manager running on the service machine at the Tier 2. With this and the usage of the condor_shared_port daemon, the necessity of opening ports in the firewall is heavily reduced. In this setup, outside daemons in need of a connection to a specific daemon in the private network connect to the connection broker and request communication with said daemon. The CCB tells this said daemon, which finally establishes the connection to the outside daemon from the inside. With this setup, only one port needs to be opened to enable all the communication between the daemons.

As CLAIX does not do whole-node scheduling, but rather places multiple jobs on the same node, special attention has to be paid to some of the configuration. First, the HTCondor daemon starting the jobs on the worker-nodes registers at the condor central manager per default only with the name of the host, it is running on. However, as sometimes multiple drones might be placed on the same host, this name is the same and thus the drones are not distinguishable in HTCondor. As a result, no drone will be running jobs, except the first to start on this host. This behavior can be avoided by explicitly setting "`STARTD_NAME`", the name of the daemon, to a unique identifier. On CLAIX, it is configured to contain the `TardisDroneUuid`, a unique ID assigned to every drone by TARDIS. Second, also the "`MASTER_INSTANCE_LOCK`" needs to be set to a unique file path. Normally it prevents starting up multiple HTCondor-instances on the same host at the same time, however, this is wanted in this case.

As CLAIX is used as a resource for CMS jobs, part of the design is running nested Singularity containers. This is not necessarily trivial. Either the container needs to be started using a user name space ("`--userns`"), or the image needs to be transformed to a "sandbox"-image. The former is only possible if user name spaces are activated in the kernel of the host. The latter transformation is recommended to be performed by root, in order for all the files to have correct permissions. For CLAIX, we contacted the cluster operations team, which kindly fulfilled our request of generating the sandbox image, also user name spaces were activated. Another obstacle with nested containers are the bind mounts in the container. Singularity sets the environment variable "`SINGULARITY_BIND`" to the list of bind mounts when invoking the container. When the second container is started within the first one, all mounts as specified in the variable, are tried to be bind-mounted again into the second container. This may fail, however, as paths within the container might have changed. The OBS HTCondor is thus configured to invoke the container with the argument "`--env SINGULARITY_BIND=␣`". This unsets the environment variable within the container.

CMS WLCG sites are expected to define configurations specific to the local site at `/cvmfs/cms.cern.ch/SITECONF/local`. This path is a softlink, which can be configured, where to point. As this was not configured in the beginning, cluster operations made it point to `/etc/cvmfs/SITECONF` as per our inquiry. Even though this location does not exist on the host, we can bind mount the specific configuration to this path inside the container. With this setup, CMS jobs encounter the configuration at the expected location.

Scaling

The setup has proven to be very successful. After a testing phase of four days, allocating 1000 cores at CLAIX, the system was scaled up to integrate 5000 cores into the Tier 2. As this is approximately the pledge to the WLCG, the number of cores for the experiments were virtually doubled for more than a week. After this time, the granted resource budget of the test-project of about seven months was exhausted and submission was blocked by the HPC cluster. The number of cores over the time span of several days can be seen in figure 2.

Remote compute element

The Aachen Tier 2 site, including the resources dynamically allocated at CLAIX, is accessible for users and the experiments through a remote HTCondor *compute element* (CE), operated by GridKa at KIT. The jobs submitted to the remote CE are passed to another local HTCondor-instance at GridKa. This instance serves as scheduler for the OBS operated at the Tier 2 in Aachen, hence the submitted jobs are run on the Aachen resources. With this setup, the computing operations team in Aachen, does not have to maintain an instance of a CE, but can

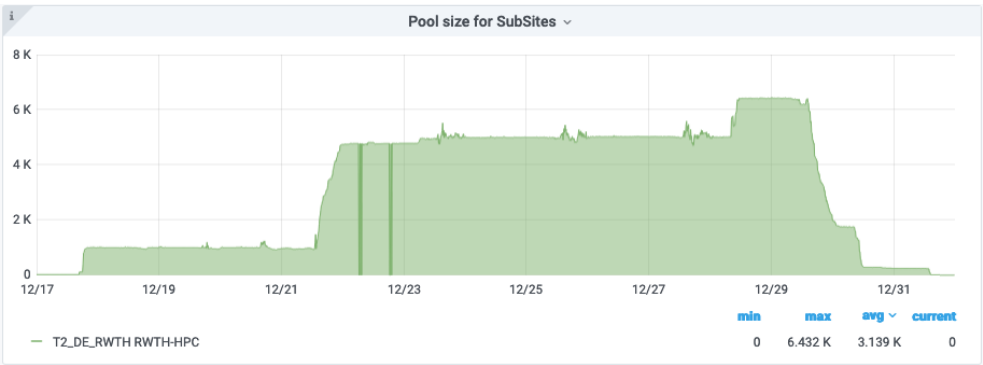

Figure 2. The number of cores, allocated at the HPC cluster CLAIX and integrated into Aachen Tier 2 over a span of 15 days. After a 4 days testing-period, the number of requested cores was increased from 1000 to 5000 cores.

rely on the CE as a service by the GridKa team. As GridKa also operates a Tier 1 WLCG site with multiple CEs, the remote CE for the Aachen Tier 2 site is managed as one of many, with minimal operational overhead.

4 Conclusion

The resource management system COBaLD/TARDIS [2, 3] enables the dynamic on demand integration of different resources. By its modular design, it allows for multiple different resource types to be integrated into one overlay batch system, which is acting as a single point of entry to the users. With a COBaLD/TARDIS-setup it was possible to transparently use 5000 cores of the non-HEP cluster CLAIX, available for local research groups, over the span of more than a week by integrating them into the Aachen Tier 2 WLCG site. The number of cores available to the experiment were thereby doubled, compared to the pledge to the WLCG of approximately 5000 cores. For providing the environment expected at WLCG resources, singularity containers were used and CVMFS repositories were bind mount into the container as needed. This setup allowed for the usage of a non-HEP specific resource, completely transparent to users and experiments through an existing WLCG site.

Acknowledgement

The authors acknowledge the support by the Federal Ministry of Education and Research of Germany (BMBF) in the project 05H18VFRC1 - "Entwicklung und Optimierung der Nutzung heterogener Rechenressourcen (Pilotmaßnahme ErUM-Data)".

References

[1] J. Albrecht, A.A. Alves, G. Amadio, G. Andronico, N. Anh-Ky, L. Aphecetche, J. Apostolakis, M. Asai, L. Atzori, et al., Computing and Software for Big Science **3** (2019)

[2] M. Fischer, E. Kuehn, M. Giffels, M. Schnepf, S. Kroboth, O. Freyermuth, *Matterminers/cobald: New plugin system* (2020), `https://doi.org/10.5281/zenodo.3752587`

[3] M. Giffels, S. Kroboth, M. Schnepf, E. Kuehn, R. Caspart, F. von Cube, M. Fischer, P. Wienemann, *Matterminers/tardis: The dead planet* (2020), https://doi.org/10.5281/zenodo.4314952

[4] I. Sfiligoi, D. Bradley, B. Holzman, P. Mhashilkar, S. Padhi, F. Wuerthwein, *The Pilot Way to Grid Resources Using glideinWMS* (2009), Vol. 2, pp. 428–432

[5] M. Fischer, E. Kuehn, M. Giffels, M.J. Schnepf, A. Petzold, A. Heiss, EPJ Web of Conferences **245**, 07040 (2020)

[6] M. Fischer, M. Giffels, A. Heiss, E. Kuehn, M. Schnepf, R.F. von Cube, A. Petzold, G. Quast, EPJ Web of Conferences **245**, 07038 (2020)

[7] D. Merkel, Linux journal **2014**, 2 (2014)

[8] G.M. Kurtzer, V. Sochat, M.W. Bauer, PLOS ONE **12**, 1 (2017)

[9] G.M. Kurtzer, cclerget, M. Bauer, I. Kaneshiro, D. Trudgian, D. Godlove, Vanessasaurus, Y. Cote, C.E.A. Gutierrez, G. Vallee et al., *hpcng/singularity: Singularity 3.7.1* (2021), https://doi.org/10.5281/zenodo.4435194

[10] P. Buncic, C.A. Sanchez, J. Blomer, L. Franco, A. Harutyunian, P. Mato, Y. Yao, Journal of Physics: Conference Series **219**, 042003 (2010)