Dynamic Computing Resource Extension using COBALD/TARDIS

R. Florian von Cube, René Caspart, Max Fischer, Manuel Giffels, Eileen Kühn, Günter Quast, and Matthias J. Schnepf

Karlsruhe Institute of Technology, Germany
{florian.cube, rene.caspart, max.fischer, manuel.giffels, eileen.kuehn, guenter.quast, matthias.schnepf}@kit.edu

Abstract. To dynamically increase computing power on demand, Cloud providers, HPC clusters, and free institute resources can be used. In order to make theses so-called opportunistic resources transparently available, the services COBALD and TARDIS are developed in collaboration of the Institute of Experimental Particle Physics (ETP) and the Steinbuch Centre for Computing (SCC) at KIT. The opportunistic resources are integrated into an overlay batch system (OBS), which acts as a single-point-of-entry for the users. Depending on the decisions of the OBS, the utilization, and the allocation of the additional compute resources, COBALD/TARDIS adjust the resource allocation at the various resource providers. To supply the necessary software environment for the jobs, required by the scientific communities, virtualization and containerization technologies are used on those heterogeneous resources.

Keywords: Opportunistic Resources · Heterogeneous Resources · Overlay Batch System · High Energy Physics · Resource Scheduling.

1 Computing Challenges in Physics

With future experiments, recording drastically increasing amounts of data, more computing resources will be needed for analysis. In those analyses, physicists want to exploit modern techniques which require specific hardware, as e.g. graphic cards. Many institutes in physics have dedicated local worker nodes which are made accessible for the end-users through a batch system. Those statically integrated resources are adapted to the physicists needs, providing the necessary soft- and hardware, but are usually provisioned for the average resource demands. With fluctuating demand in computing power, this can result in long waiting times during peak loads.

As the landscape in which scientific computing takes place shifts from community-specific dedicated computing clusters, to shared science computing centers, HPC clusters, and cloud providers, already today, those can be used for

mitigating load peaks or providing specific hardware. By backfilling with jobs from other communities, those resources can achieve higher efficiencies. The usage of the shared clusters, however, is often inconvenient for the end-user and requires a lot of effort, because of multiple identity providers, the need to provision software and each cluster operating its dedicated workload management system. With several such resources, the end-user has to assess which the most suitable resource is and check which resource has capacity at the current time. If a resource is only temporarily available, this decision gets even more challenging.

2 Single point-of-entry

For the end-user, usage through a single point-of-entry with dynamic integration and provisioning of a heterogeneous resource-pool, containing all resources, also the opportunistic ones, is more desirable. To enable a transparent usage, the opportunistic resources are integrated into an overlay batch-system (OBS). This can be e.g. the batch system of the institute or the collaboration. By integrating the opportunistic resources into the OBS, the end-user only has to interact with this layer, instead of with each resource separately. Authentication and authorization at the several opportunistic resources is done through proxy users per institute or research group. With this further authentication of the end user becomes dispensable.

The OBS also takes care of the decision of the job to resource scheduling and accounting. However, providing the user with a single point-of-entry to a plethora of heterogeneous resources, it has to be taken care, that the jobs runtime environment is the same, no matter which resource it is scheduled to. For this, containerization and virtualization technologies are used. With those, it is assured, that e.g. specific software is installed.

3 Resource Integration using TARDIS

The integration of the opportunistic resources is performed by TARDIS [11] (Transparent Adaptive Resource Dynamic Integration System). TARDIS starts so called DRONEs on the opportunistic resource, which integrate the resource into the OBS. A DRONE is a placeholder job that allocates the resource and integrates it in the OBS. The DRONE runs either natively on the resource, or it is started in a container, or a virtual machine. The DRONE is responsible for preparing the resource to accept jobs from the OBS. When this preparation is performed, the OBS can place jobs on the newly integrated resource.

3.1 Overlay Batch-System Adapter

For the integration of the opportunistic resources into the OBS, TARDIS interfaces the OBS through a `BatchSystemAdapter`. The adapter defines, how resources are added to the OBS and how they are managed. At the moment

TARDIS supports HTCondor [12] as OBS. An adapter for the Slurm Workload Manager is currently under development in collaboration with the university of Freiburg.

3.2 Resource Provider Interface

For the life cycle management of the opportunistic resources, they are interfaced with `SiteAdapter`s. A `SiteAdapter` defines, how the resource is allocated, how the status of the drone can be monitored, and how drones can be terminated.

At the moment HTCondor, MOAB [6], and Slurm [15] batch systems, as well as CloudStack [7], and OpenStack [14] cloud APIs are supported.

4 Resource Mix Assessment with COBalD

To optimize the usage of the integrated opportunistic resources, the decisions of the OBS are monitored by assessing the suitability of the presently integrated resources to the current job mix. This assessment is done by COBalD [9] (COBalD – the **O**pportunistic **Bal**ancing **D**aemon).

For each resource component like CPU, memory, or disk space, the ratio of the assigned and the requested component is determined. Here, "requested" means requested by the drone, and "assigned" is what is assigned to the actual jobs running on the resource. The maximum and minimum are called *allocation* and *utilization*, respectively:

$$\mathit{allocation} = \max\left(\frac{\mathrm{CPU}_{\mathrm{assigned}}}{\mathrm{CPU}_{\mathrm{requested}}}, \frac{\mathrm{Memory}_{\mathrm{assigned}}}{\mathrm{Memory}_{\mathrm{requested}}}, \cdots\right)$$
$$\mathit{utilization} = \min\left(\frac{\mathrm{CPU}_{\mathrm{assigned}}}{\mathrm{CPU}_{\mathrm{requested}}}, \frac{\mathrm{Memory}_{\mathrm{assigned}}}{\mathrm{Memory}_{\mathrm{requested}}}, \cdots\right).$$

By comparing those, a suitability of the drone to the current job mix can be determined. Consider the following example as shown in fig. 1: A drone requesting a virtual machine with 20 CPU-cores and 80 GB of memory at the resource provider, runninng jobs which sum up to 20 CPU cores, but only use 60 GB of memory. With this the machine is fully allocated as all CPU-cores are assigned and it can not run any more jobs. However the memory is only utilized to 75%. This implies that the machine is not the optimal fit for the current job mix.

Depending on the *utilization* and the *allocation* of the drones, COBalD determines the *demand* for the respective resource. That is the number of drones of that type that should be requested. On the other hand *supply* is the number of drones of the same type, currently available in the OBS. The policy, when to request more drones of a specific type is configurable with a `Controller`. It increases the *demand*, when drones of the specific type have a high *allocation* and decreases it if the *utilization* is low.

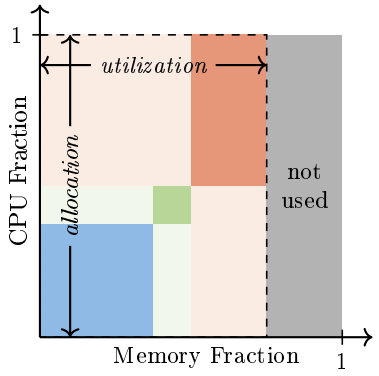

Fig. 1. Depiction of the metrics "*allocation*" and "*utilization*" used by COBALD to assess the suitability of the DRONE to the presently running jobs. The jobs are shown as the darker rectangles. The gray area is a resource component fraction, not used by any job. The values measured are directly influenced by the decisions of the overlay batch system and thus can be used for reacting accordingly.

COBALD uses TARDIS resource provider interfaces to determine the number of DRONEs which should be requested at each one of them. An overview of the resource scheduling process, depending on the decisions of the OBS is shown in fig. 2. Because of their modularity COBALD/TARDIS are easily customizable and can be adapted to virtually every setup.

5 Examplary Setups

COBALD/TARDIS is used for resource scheduling in production environments in several institutes within and outside of physics.

5.1 Institute of Experimental Particle Physics

At the Institute of Experimental Particle Physics (ETP) of the Karlsruhe Institute of Technology (KIT), research groups of for example the AMS, Belle, and CMS collaboration share one local HTCONDOR batch system. For data analysis and simulation, multiple resources are used:

- The normal desktop workstations, when not used interactively, and dedicated local worker nodes with 600 CPU-cores are integrated in the OBS. All jobs run in DOCKER-containers [8].
- The HPC cluster FORHLRII at KIT with 25.000 CPU-cores is integrated using COBALD and TARDIS. The DRONEs are bash-scripts, backfilled to the SLURM Workload Manager, which start up a HTCONDOR-process. The jobs run in a SINGULARITY-containers [13].

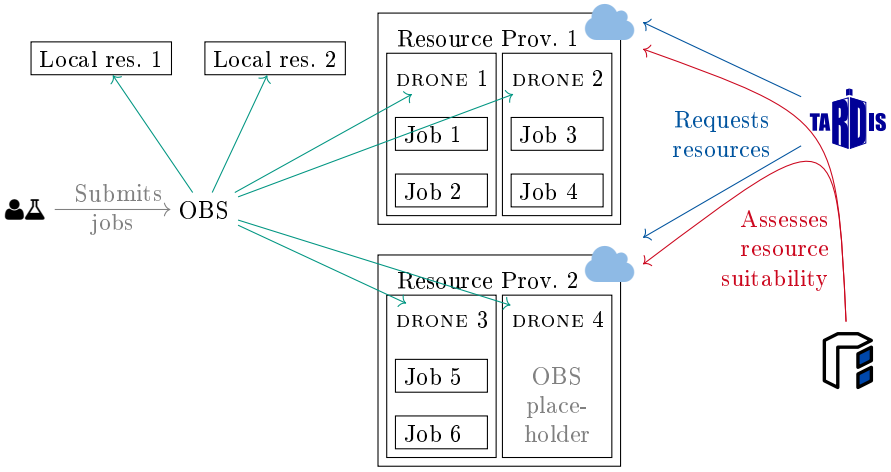

Fig. 2. Jobs are submitted to the overlay batch-system (OBS). The OBS decides, on which DRONE to run the jobs. Depending on the metrics *allocation* and *utilization* of the resources, COBALD manages the amount and type of resources needed. In the depicted example, more resources would be requested at resource provider 1.

– The HPC cluster BWFORCLUSTER "NEMO" at the university of Freiburg is also integrated using COBALD and TARDIS. It is shared with several other communities and consists of 900 worker nodes summing up to roughly 20.000 CPU-cores. The DRONEs run as virtual machines, booted through OPENSTACK via the MOAB batch system. Within the virtual machines, provided by the HPC center, the jobs run inside of DOCKER-containers.

All resources are available to the physicists at ETP completely transparent through the batch system. With this setup several thousand CPU-cores are opportunistically integrated and used as shown in fig. 3.

5.2 WLCG Tier 1 Center GridKa

GridKa is one of the 13 tier 1 centers of the Worldwide LHC Computing Grid (WLCG), located at KIT. It provides compute and storage resources for several high energy and astroparticle physics collaborations. As shown in fig. 4 opportunistic resources are transparently made available to the collaborations through one dedicated compute entry-point, a so-called compute element (CE). The resources are managed using COBALD and TARDIS. This also allows for dynamic integration of dedicated hardware, as e.g. graphic cards for deep learning.

5.3 Fighting COVID-19

The Institute of Experimental Particle Physics and the tier 1 center GridKa contribute to the distributed computing projects "Rosetta@home" [4] and "Fold-

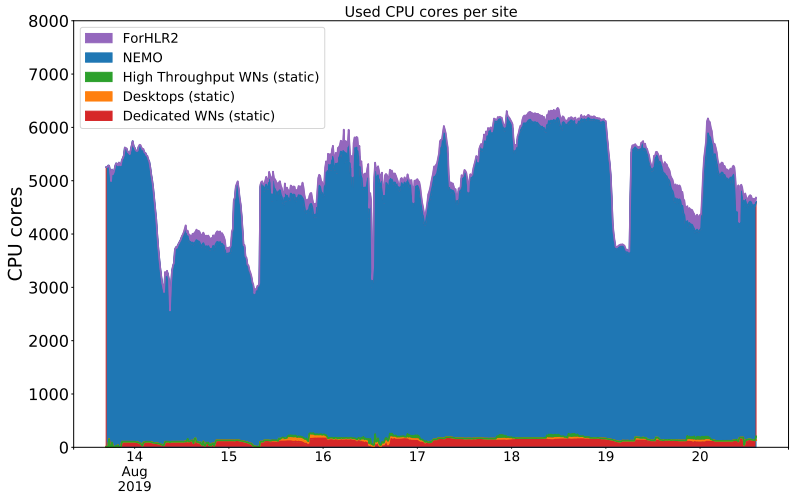

Fig. 3. The number of CPU cores used through the OBS at ETP. The resources at the HPC clusters NEMO and ForHLRII are dynamically integrated using COBaLD and TARDIS. The number of the statically integrated CPU cores used fluctuates, as those are freed, whenever the machine is used interactively.

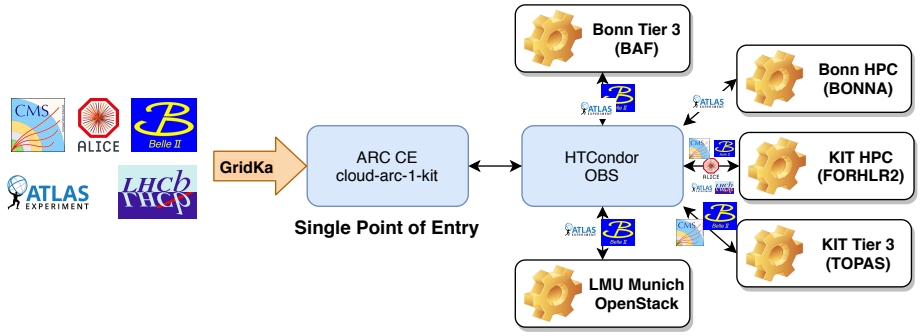

Fig. 4. GridKa dynamically integrates opportunistic resources for several experiments behind a single point-of-entry. Jobs from the different experiments are only dispatched to their corresponding resources. This happens completely transparent to the experiments. Figure from [10].

ing@home" [2] for the fight against COVID-19. Both institutes use a COBALD/TARDIS-setup with a placeholder batch-system mimicking a constant demand to allocate resources within their respective resource pools for providing them to the protein research projects. The computing payloads are pulled from the central management servers of the research projects and then executed on the local resources. With this setup, ETP and GridKa make a significant contribution. [3,5,1]

6 Summary

Using shared science computing centers, HPC clusters, and resources at cloud providers, is a promising way to mitigate computing load peaks within research groups, and provide specific hardware. It also can help to achieve higher efficiencies on those resources. To dynamically schedule and integrate heterogeneous resources on demand, COBALD and TARDIS are the necessary toolkits. They allow for optimizing the resource usage by reacting on the decisions of the overlay batch-system and by this yield a more efficient resource usage.

For the end-users a setup with COBALD and TARDIS allows for a transparent usage of a plethora of resources through one single point-of-entry. They are no longer faced with the inconvenience and the effort of having multiple identity providers, or several batch systems to deal with. By integrating resources providing dedicated hardware, this can easily be made available for the end-users maintaining fair usage.

References

1. Fight COVID-19 – Grafana (2020), https://grafana-sdm.scc.kit.edu/d/uMJtTojZk/fight-covid-19
2. Folding@home – Fighting disease with a world wide distributed super computer. (2020), https://foldingathome.org/
3. Folding@home stats report (2020), https://stats.foldingathome.org/team/250565
4. Rosetta@home (2020), https://boinc.bakerlab.org/
5. Rosetta@home: KIT-GridKa (2020), https://boinc.bakerlab.org/rosetta/show_user.php?userid=2127744
6. Adaptive Computing: Moab HPC – Adaptive Computing (Jun 2020), https://adaptivecomputing.com/cherry-services/moab-hpc/
7. Apache: Apache CloudStack: Open Source Cloud Computing (Jun 2020), http://cloudstack.apache.org/
8. Docker Inc.: Empowering App Development for Developers | Docker (Jun 2020), https://www.docker.com/
9. Fischer, M., Kuehn, E., Giffels, M., Schnepf, M., Kroboth, S., Freyermuth, O.: MatterMiners/cobald: New Plugin System (Apr 2020). https://doi.org/10.5281/zenodo.3752587, https://doi.org/10.5281/zenodo.3752587
10. Giffels, M., et al.: Effective dynamic integration and utilization of heterogenous compute resources. Proceedings of the 24th International Conference on Computing in High Energy and Nuclear Physics (2020), to be published

11. Giffels, M., Schnepf, M., Kuehn, E., Kroboth, S., Caspart, R., von Cube, R.F., Fischer, M., Wienemann, P.: MatterMiners/tardis: YAR (Yet Another Release) (Jun 2020). https://doi.org/10.5281/zenodo.3874847, https://doi.org/10.5281/zenodo.3874847
12. HTCondor Team: HTCondor (May 2020). https://doi.org/10.5281/zenodo.3834815, https://doi.org/10.5281/zenodo.3834815
13. Kurtzer, G.M., Sochat, V., Bauer, M.W.: Singularity: Scientific containers for mobility of compute. PLOS ONE **12**(5), 1–20 (05 2017). https://doi.org/10.1371/journal.pone.0177459, https://doi.org/10.1371/journal.pone.0177459
14. OpenStack: Build the future of Open Infrastructure (Jun 2020), https://www.openstack.org/
15. SchedMD: SchedMD | Slurm Support and Development (Jun 2020), https://www.schedmd.com/