EPJ Web of Conferences **251**, 02039 (2021) https://doi.org/10.1051/epjconf/202125102039
CHEP 2021

Transparent Integration of Opportunistic Resources into the WLCG Compute Infrastructure

Michael Böhler[3], *René* Caspart[1,*], *Max* Fischer[1], *Oliver* Freyermuth[2], *Manuel* Giffels[1], *Stefan* Kroboth[3], *Eileen* Kuehn[1], *Matthias* Schnepf[1], *Florian* von Cube[1], and *Peter* Wienemann[2]

[1]Karlsruhe Institute of Technology (KIT), Germany
[2]University of Bonn, Germany
[3]Physikalisches Institut, Albert-Ludwigs-Universität Freiburg, Freiburg, Germany

Abstract. The inclusion of opportunistic resources, for example from High Performance Computing (HPC) centers or cloud providers, is an important contribution to bridging the gap between existing resources and future needs by the LHC collaborations, especially for the HL-LHC era. However, the integration of these resources poses new challenges and often needs to happen in a highly dynamic manner. To enable an effective and lightweight integration of these resources, the tools `COBalD` and `TARDIS` are developed at KIT.
In this contribution we report on the infrastructure we use to dynamically offer opportunistic resources to collaborations in the World Wide LHC Computing Grid (WLCG). The core components are `COBalD/TARDIS`, HTCondor, CVMFS and modern virtualization technology. The challenging task of managing the opportunistic resources is performed by `COBalD/TARDIS`. We showcase the challenges, employed solutions and experiences gained with the provisioning of opportunistic resources from several resource providers like university clusters, HPC centers and cloud setups in a multi VO environment. This work can serve as a blueprint for approaching the provisioning of resources from other resource providers.

1 Introduction

The Worldwide LHC Computing Grid (WLCG) has historically been built on a homogeneous computing environment, relying on member sites to provide a common base functionality. Today, the spread of modern technologies and generalization of usage models, such as cloud computing or idle cycle scavenging, means there is a significant volume of compute resources outside the permanent member sites of the WLCG. Even though the underlying technologies exist for years, the proper usage of such *opportunistic resources* is still the focus of ongoing research and development: the heterogeneity as well as the dynamicity require capabilities well beyond those of resource management systems aimed at permanent, homogeneous resources.

In this paper, we present an approach for opportunistic resource management that allows to combine per-resource-provider expertise with uniform, transparent resource access. The major contribution is the demonstration of scalability and real-world feasibility of previous

*e-mail: Rene.Caspart@kit.edu

prototypical work applied to production usage at national scale. For this, we present the high-level idea of our approach, several case studies of opportunistic resource providers, as well as the evolution into a unified setup allowing for monitoring and accounting.

The fundamental idea of our work is to maximize transparency for all involved parties: complexities of each task are handled by the respective experts, without significantly impacting other tasks. Conversely, our approach should be able to scale well beyond even what we have achieved so far.

2 Opportunistic Resource Federation at National Scale

Providing transparent access to resources from multiple sites or resource providers can be divided into several subtasks: First, available resources should be presented via a single access point for transparency towards resource users. Second, required resources should be automatically acquired for transparency towards resource providers. Finally, accessing resources should provide a common baseline of functionality to bridge the expectation gap between resource users and providers. The overarching goal of these subtasks is a strong separation of concerns: resource providers do not need to adapt to resource users, and vice versa.

Previous and related work in the context of WLCG generally follows the same basic structure: Resources are aggregated in a single *Overlay Batch System* (OBS), with a *meta-scheduler* predicting which resources must be acquired, using *pilot jobs* to allow access from the OBS to resources. While such approaches are proven to scale well for homogeneous resource providers, the high diversity of opportunistic resources and provider-centered scope necessitates modifications.

2.1 Single point of entry and the intermediate OBS

The scope of existing OBS in the WLCG is tied to user groups, not resource providers: an OBS represents all resources acquired by the Workflow Management System of a collaboration [1–3]. While integrating opportunistic resources into an existing OBS makes resources directly accessible to users, it also restricts access to the specific group owning the OBS. For example, the meta-scheduler Cloud Scheduler [4] directly integrates resources into the distributed job submission infrastructure of specific collaborations.

In order to benefit from the advantages of an OBS, namely extensibility with and transparency of resource, without restriction to a specific user group, we introduce an *intermediate OBS*. The intermediate OBS aggregates resources at the scope of several resource providers; at the same time it acts like a regular, multi-group batch system towards users. Notably, this usage model still allows resources to be late-bound by another OBS: the pilot of the OBS is transparently executed by the intermediate OBS on the opportunistic resource. The meta-scheduler HEPCloud [5] follows a similar approach utilizing an OBS for scaling, but still restricting access to a specific user group.

In practice, we have realized such an intermediate OBS on a national scale using established services as already used regularly in the WLCG (see Fig.1): An instance of the HTCondor batch system, which has been proven to scale well beyond our needs [6], aggregates the resources acquired from resource providers. Grid Compute Elements (CE) provide a point of entry that is indistinguishable from those of regular WLCG batch systems. On the one hand, this minimizes the integration efforts needed by collaborations – our intermediate OBS and thus its resources are accessed exactly like existing WLCG resources. On the other hand, this minimizes the operational effort – the intermediate OBS and CE are deployed at the GridKa Tier 1, using the same configuration management templates as for the Tier 1 batch system and CE.

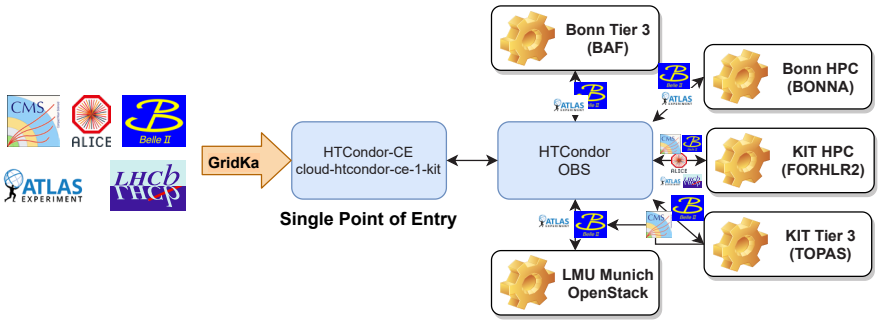

Figure 1. Schematic view of the job flow from the experiments to the integrated opportunistic resources via the Grid CE and the intermediate OBS.

Even though an intermediate OBS simplifies the aggregation and access to opportunistic resources, it also complicates the meta-scheduling of these resources: The simplest use case layers three batch system on top of each other, two of which use late-binding for their resources.[1] Whereas resource requirement prediction given directly late-bound payloads is challenging but possible [7], we found prediction to be unfeasible given the latency and multiple layers of late-binding.

2.2 Provider specific resource management using `COBalD/TARDIS`

To tackle the challenges of meta-scheduling in complex environments, the `COBalD/TARDIS` software suite [8, 9] primarily developed at Karlsruhe Institute of Technology aims at an evolution of proven approaches and a revolution of challenges: We adapt the proven idea of having a single software framework providing a unified abstraction over common resource provider technologies; however, we evolve the idea of *pilots* as a concrete implementation to more general *drones* as a framework itself. Most importantly, we abandon the idea of globally unified resource prediction for all jobs and resource providers. The following is a brief summary of the core concepts as relevant for this paper, an in-depth discussion being available in previous publications [10, 11].

The major issue with global resource prediction is that it must anticipate the decisions of all schedulers with incomplete information: these schedulers are black-boxes, any late-bound jobs are not known ahead of time, and the latency from all scheduling layers puts a hard requirement on the minimum look ahead. In practice, this is difficult for even one of multiple schedulers, late-binding or significant look ahead [12]. Thus `COBalD` – the `O`pportunistic `Bal`ancing `D`aemon – uses a different approach based on reaction instead of prediction: each `COBalD` instance monitors how well its resources are used and subsequently provisions more well-used resources or deprovisions unused resources. Notably, this means that multiple `COBalD` instances can trivially serve the same (intermediate) OBS.

`TARDIS` – the Transparent Adaptive Resource Dynamic Integration System – is a plugin to `COBalD`, providing production ready adapters for common batch systems and resource provision technologies. This includes both the tasks of provisioning/deprovisioning resources, as well as integrating them into an intermediate OBS. The `TARDIS` equivalent of pilots, dubbed drones, focuses on semi-autonomous resource integration; a drone is agnostic to the specific

[1] More generally, for a setup involving n layers, $n - 1$ layers are late-bound

needs of the WLCG, and can be fully customized to match its target environment and use case.

In combination, the COBalD/TARDIS suite is a production ready suite to provision resources from individual providers for an (intermediate) OBS. As several instances can operate in parallel, it is possible and advisable to deploy a separate instance per resource provider or even per resource flavor of each provider. This allows to adjust each instance to the requirements of its specific resource provider, combining general templates with expertise of local experts.

3 Integration of Opportunistic Resources

Leaving the scope of homogeneous resources provided by WLCG and heading towards a more heterogeneous environment where each resource is unique and different, poses additional challenges to be tackled. At this point it is reasonable to use synergies as far as possible in order to reduce the overhead required for each integration to a minimum.

This applies in particular to the provisioning of the WLCG like software environment expected to be present by the experiments. For all resources described below either traditional virtual machines or modern container technologies like Singularity [13] or Charliecloud [14] are used to provide the mandatory WLCG worker node environment. The virtual machines/containers are based upon official Centos 7 images supplemented by additional software packages like the HEP_OSlibs [15] and worker node [16] meta-packages. The built containers are then stored in unpacked form on a CVMFS file system [17], which is either inherently mounted on the compute nodes itself or provided by the `cvmfsexec` [18] tool allowing for unprivileged mounts of CVMFS repositories in user space. Furthermore, CVMFS is also used to provide the experiment software as well as parts of the Grid middleware.

In addition, it is also advisable to repurpose existing edge services like Frontier [19] and CVMFS Squid proxies, which are fortunately operated close by to the integrated resources described below.

3.1 University of Bonn

At the University of Bonn we explored two different scenarios: Integration of a Tier 3 cluster to which the authors have administrative access and exploitation of HPC resources run by external operators. Although demands are quite different for those use cases, we tried to use as much synergy as possible. To integrate both clusters we run a common COBalD/TARDIS node with two instances of the service as interface between the intermediate overlay batch system and the local batch systems. The machine and service configuration are deployed fully automated using the local Foreman/Puppet [20, 21] infrastructure. To configure and control the COBalD and TARDIS services a dedicated Puppet module has been developed [22].

To achieve a maximum of job mobility we launch drones inside containers. We use official CentOS 7 images from Docker Hub [23] as described above, augment it by configuration adjustments and add an HTCondor execute node setup which is configured in such a way that it integrates itself into the intermediate OBS. This image is rebuilt either if configuration changes are pushed to its repository or at least once a day to ensure the latest security and bug fix updates are included.

This setup is presently used to run jobs from the ATLAS and the Belle experiments at both clusters at the University of Bonn.

3.1.1 Bonn Analysis Facility

The Bonn Analysis Facility (BAF) [24] is run by Physikalisches Institut at the University of Bonn. It serves – among others – as an ATLAS tier 3 site and provides compute resources to a wide range of other local physics research groups. The diversity of research fields comes along with a significant variation of compute and storage requirements. To cope with the multitude of different software stacks more easily and to decouple the operating system used to run the underlying infrastructure from the user software stack, all jobs are run inside Singularity [13] containers on this cluster. This was implemented using HTCondor's integrated container support. Therefore the only difference between local user jobs and jobs deployed via `COBalD/TARDIS` is that a different container flavor is chosen. Accessing the container images on the compute nodes is also straightforward since CVMFS clients are running on all worker nodes and all necessary repositories are bind-mounted inside the containers.

The drone jobs are submitted to the local HTCondor batch system directly from the `COBalD/TARDIS` node, which acts as a regular submit node using Kerberos to authenticate to the local batch system as a VO-specific service user. The users are handled by regular fair-share, but given a tunable weight: at the time of writing, their priority is decreased by a factor of 20 as compared to regular users, effectively performing backfilling. A high core quota set in the `COBalD/TARDIS` configuration effectively allows backfilling of the full cluster, while the drones are automatically drained after 12 hours of lifetime to ensure sufficient dynamics for regular user jobs to take over.

3.1.2 Bonna High Performance Computing Center

Bonna is the central HPC cluster at the University of Bonn. It is run by the Fraunhofer-Institut für Algorithmen und Wissenschaftliches Rechnen (SCAI) on behalf of the university. This cluster neither offers integrated container support nor CVMFS clients on its worker nodes. Therefore a slightly different approach had to be chosen to run `TARDIS` drones on Bonna. Drone submission is performed remotely via SSH to the Bonna Slurm batch system. To access the necessary CVMFS repositories, the tool `cvmfsexec` is used. It allows unprivileged users to mount CVMFS repositories provided that some prerequisites are met. The situation on Bonna at the time of writing is that `fusermount` and unprivileged user namespaces are available. This allows `cvmfsexec` to create a mount namespace in which the desired CVMFS repositories are accessible via the usual `/cvmfs` mount point. Subsequent commands are run in this environment. In our case we launch Charliecloud containers using the same unpacked images from the CVMFS infrastructure which we also use on BAF. After the commissioning of this setup `cvmfsexec` learned to directly start Singularity containers from CVMFS. Still we stuck to Charliecloud which has been designed as lightweight container solution to be used by unprivileged users.

A disadvantage of using `fusermount` is that it is difficult to ensure that all mount points created by `cvmfsexec` are properly cleaned up on job exit. To overcome these shortcomings, `cvmfsexec` offers making use of unprivileged namespace FUSE mounts. In this case the mounts are performed in an additional process ID (PID) namespace which makes sure that the kernel cleans up mount relics upon job termination. Unfortunately the presently used kernels on Bonna are too old to support this feature[2] but we hope that this will be remedied soon.

[2]It requires at least kernel 4.18 as it is available e.g. on CentOS 8 but it has also been backported to the CentOS 7 series starting from CentOS 7.8.

3.2 Karlsruhe Institute of Technology

Similar to the situation at the University of Bonn, at Karlsruhe Institute of Technology two compute clusters are integrated in the opportunistic infrastructure: on the one hand the HEP specific local university cluster, the *Throughput optimized Analysis System* (TOPAS) [25] and on the other hand the HPC cluster *Forschungshochleistungsrechner ForHLR Phase II* (ForHLR II) [26]. Both of these resources are integrated in a backfilling mode, meaning other usages take precedence and only otherwise idling resources are provisioned. The resources at ForHLR II are available to all collaborations supported by the local Tier 1 WLCG center GridKa, while the usage of the resources at TOPAS is limited to the CMS and Belle II collaborations.

To simplify the setup at KIT as many synergies as possible are exploited for the integrations of both clusters. Drones for both clusters utilize HTCondors built-in support for singularity to provide the basic worker node environment expected by the WLCG collaborations. The configurations for the HTCondor daemons required by the drones are stored in a common git repository and pulled in using `condor-git-config`, a custom tool utilizing HTCondor's "include command" functionality [27]. Due to the close proximity to GridKa, the setup for both clusters utilizes edge services provided by the Tier 1 center.

3.2.1 TOPAS

The TOPAS cluster is a local university Tier 3 cluster designed for high-throughput HEP analyses. Both the TOPAS cluster and the intermediate OBS use HTCondor as workload management system. To ease the deployment of drones at the TOPAS cluster, the locally available HTCondor binaries are used for the integration in the intermediate OBS. While given its primary purpose for running local HEP user analysis workflows, the setup of the TOPAS worker nodes complies with the basic environment expected by the WLCG collaborations, this environment is nonetheless provided via a singularity container. This further allows utilizing the setup at the TOPAS cluster, where the authors have administrative access, as a test bed for changes to both the TOPAS and ForHLR II setup.

In a recent extension to the TOPAS cluster additional GPU nodes have been deployed. To allow for the utilization of these GPUs via the intermediate OBS, drones requesting and providing GPUs are deployed. The batch system and GPU drones utilize the standard HTCondor mechanism for providing and requesting GPU resources. The association of drones with GPU devices is achieved using environment variables, which are propagated through the deployment layers. Due to limitations in the singularity layer a modified version of the image providing the basic worker node environment is required. This modified image includes the necessary Nvidia drivers and libraries needed for the GPU drones. The modification of the image is performed using functionalities provided by Charliecloud, allowing to integrate the required files into the image sandbox. Since this introduces a dependency on the setup of the host system, the resulting image is distributed to the GPU worker nodes using a local shared file system. At the time of writing integrated GPU resources from the TOPAS cluster are being used by the CMS Collaboration.

3.2.2 ForHLR II

The ForHLR II is a HPC cluster operated by the Steinbuch Centre for Computing at KIT. The worker nodes at the ForHLR II cluster are set up to provide support for singularity and user namespaces. The required CVMFS repositories are provided using `cvmfsexec`. The cache for CVMFS is setup as an alien cache shared among all clients hosted on the parallel

file-system of the HPC cluster. Since alien caches for CVMFS are unmanaged, the caches are rotated and cleaned up on a monthly basis thereby avoiding clogging of the available disk-space.

The drones at ForHLR II are submitted as jobs to the local Slurm workload manager via SSH. The HTCondor daemons required for the drones are run bare-metal on the Red Hat Enterprise Linux 8 worker nodes using the pre-compiled binaries provided by the HTCondor Team. Due to limits enforced in the local Slurm setup, only 50 jobs of a user may run at a time. As the drones are designed to fully utilize a single worker node, this limits the maximum number of provided CPU cores to 1200.

In 2021 the successor of the ForHLR II HPC cluster will be commissioned at KIT. The HoreKa cluster[28] will in additional to x86 CPU resources also provide Nvidia A100 GPU resources. Due to the similarity in the setup of both HPC clusters we expect to be able to integrate the CPU resources of the new cluster with little adaptions. In addition, profiting from the experience gained with integrating GPU resources at the TOPAS cluster and due to the similarities of the drone setups, we expect to be able to integrate the available GPU resources with little additional effort.

3.3 Leibniz Supercomputing Centre Garching

The Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities (LRZ) in Garching operates a cloud computing service [29] based on Open Stack [30], has been integrated into WLCG compute infrastructure using `COBalD/TARDIS` and provides additional compute resources for the ATLAS and Belle II experiments. Due to the limited amount of available resources (40 CPU cores), this setup has to be considered more as a proof of concept operated in a production environment.

The resource management is driven by a `COBalD/TARDIS` instance remotely operated at KIT as one of many in order to reduce the integration efforts needed by the local Munich physics groups. The resources comprise traditional virtual machines providing the WLCG worker node environment. In addition, HTCondor is installed inside the image and dynamically configured using `condor-git-config` in order to join the intermediate OBS using the HTCondor pool password method. Due to the proximity to the WLCG Munich Tier 2 Grid Centre also operated at LRZ, edge services are repurposed for the LRZ cloud resources.

3.4 Wrap-up on integrated opportunistic resources

All together the amount of resources opportunistically contributed to the WLCG in 2020 is shown in Fig. 2 on a monthly basis per resource provider. In total the experiments could profit from additional 7.8 million core hours.

4 Monitoring and Accounting

Complex interconnected infrastructure with multilayered batch systems such as the ones described in this paper involve many components and various interfaces. Network connections can be unreliable and it is not uncommon to observe unpredictable temporal behavior, such as delayed action following a request. Issues arising in such an environment can be difficult to debug, particularly when the individual components of the infrastructure are managed by dedicated teams. Continuous and centralized monitoring of relevant metrics at a sufficient resolution which is accessible by all involved parties is therefore essential. Metrics typically collected are the utilization of system resources and network traffic as well as general system

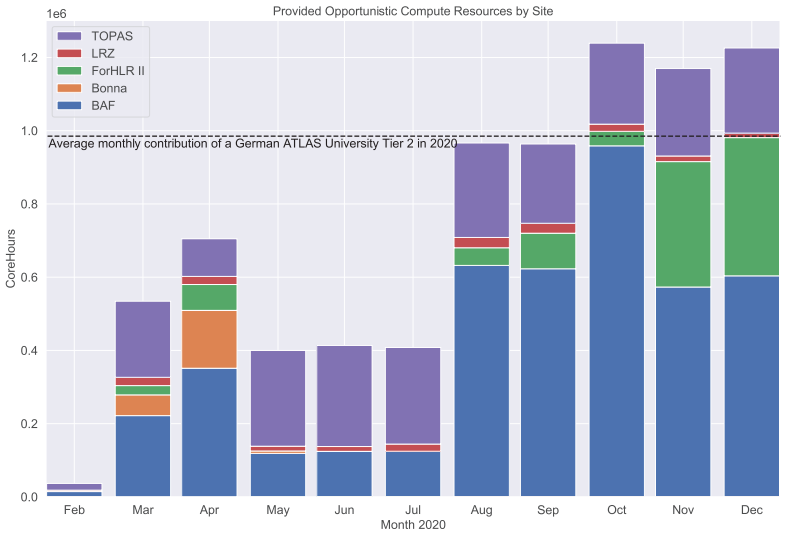

Figure 2. Additional resources measured in core hours supplied by opportunistic resource providers in 2020. The dashed line indicates the average resource contribution of an average German ATLAS University Tier 2 in 2020 [31].

and service health information, which can be obtained from the nodes running the services and the drones. In addition, the involved software and services such as `COBalD/TARDIS` and batch system schedulers expose metrics which can be vital for the assessment of system health. `COBalD/TARDIS` offers three monitoring plugins which can export information about the current state of the drones to the data stores Telegraf [32], Prometheus [33] and Elasticsearch [34]. These plugins export the current number of drones in each state as a time series or every state change of a drone as an event. The data stores are specialized in the way they store and access the data. The time-series database Prometheus regularly fetches the data from a source while Telegraf follows a push model. Event-like data is stored in Elasticsearch, which also offers a fast and powerful search engine to access the data. The wide range of data endpoints ensures that the appropriate storage is used depending on the type of data. Grafana is used for visualization as it offers means for aggregating metrics from various sources which facilitates system performance tuning and the identification of potential root causes. An example of how such aggregated data can provide valuable insights is depicted in Fig. 3 which illustrates the dynamics of how the number of drones increases after a large number of jobs are submitted to the OBS. Fig. 4 visualizes the monitoring infrastructure. To ease the deployment of the monitoring infrastructure a highly configurable Puppet module is in development.

When incorporating opportunistic resources into an OBS, it is vital to provide a mechanism which properly accounts for the consumed resources. For instance, a site might have to fulfill a pledge to an external entity; however, if the site aims to fulfill its pledge by having its computing resources opportunistically integrated into another cluster, it is not able to directly report its provided resources. In such a setting, an accounting system acts as a means to separate the information about the provided resources by site/cluster again. Based on the interfaces developed for the monitoring infrastructure, it is planned to develop an accounting system which collects the relevant data, particularly about the drones, and stores them in a

dedicated database. These data include for instance the run time and the consumed resources (CPU, RAM, Disk, ...) of the drone as well as the cluster where it was running on. The architecture of the accounting service will include an interface which provides other services or plugins with access to the data. This allows for the implementation of plugins which handle the interaction with external accounting systems like the central accounting system of the WLCG. In addition, specialized plugins tailored for specific applications can be implemented, such as inter-site billing systems or software which manages fair-shares across clusters (Fig. 4).

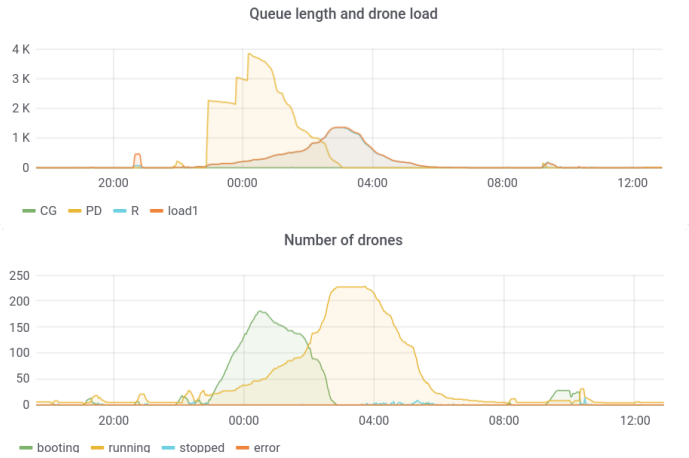

Figure 3. Monitoring both the queue length of the batch system (PD = pending, R = running) and the number of drones in their states enables one to assess how quickly the setup reacts to the sudden submission of a large number of jobs.

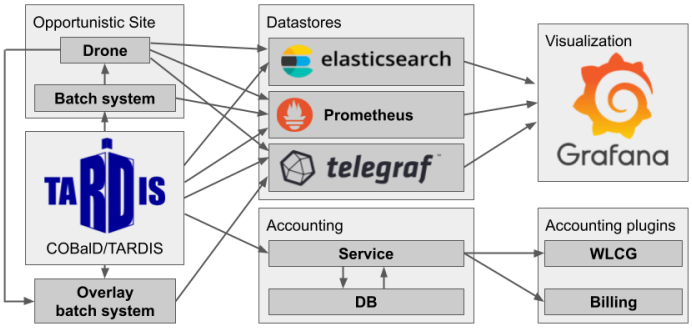

Figure 4. Illustration of the monitoring/accounting infrastructure. `COBalD/TARDIS` as the core component provides information about the state of the infrastructure. Data is also collected from the batch system as well as the drones.

5 Summary & Outlook

We have demonstrated the scalability and real-world applicability of our approach to integrate opportunistic resources into the WLCG compute infrastructure. During that work, the

used tool set has been evolving from the prototype stage to production usage at national scale. Through a collaborative effort between the University of Bonn, the KIT and the University of Freiburg five diverse compute resources – each a unique challenge in and of itself – could be integrated in a nationwide federated infrastructure delivering in total 7.8 million additional core hours to the WLCG experiments in 2020. By re-using existing and well established technologies like Grid CEs as entry point and an intermediate OBS as unified resource pool the integration efforts needed by the experiments have been minimized, while the operational effectiveness on the provider side could be increased. The dynamic provisioning/deprovisioning as well as the transparent integration of resources into the intermediate OBS is handled by the `COBalD/TARDIS` resource manager taking into account the actual resource utilization and aiming to increase the efficiency.

For the future we envisage to develop tools to improve the monitoring capabilities and to enable per provider accounting of the resources in the APEL accounting portal [35] used withing the WLCG infrastructure. Heading towards enabling fair accrediting of contributed opportunistic resources from participating providers by the experiments in order to acknowledge the efforts made and to motivate more to join.

6 Acknowledgements

The authors are grateful for being able to use the Bonna cluster at the University of Bonn. In addition, parts of this work is using the supercomputer ForHLR II funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the Federal Ministry of Education and Research.

Parts of the work were supported by the Federal Ministry of Education and Research (BMBF) within the project 05H18VFRC1 - "Entwicklung und Optimierung der Nutzung heterogener Rechenressourcen (Pilotmaßnahme ErUM-Data)". The monitoring code was mainly developed and validated on the HPC-cluster NEMO in Freiburg, which is supported by the Ministry of Science, Research and the Arts Baden-Württemberg through the bwHPC grant and by the German Research Foundation (DFG) through grant no INST 39/963-1 FUGG.

References

[1] I. Sfiligoi, D.C. Bradley, B. Holzman, P. Mhashilkar, S. Padhi, F. Wurthwein, WRI World Congress **2**, 428 (2009)

[2] P. Nilsson, J. Caballero, K. De, T. Maeno, M. Potekhin, T. Wenaus, Proceedings of XII Advanced Computing and Analysis Techniques in Physics Research, Erice, Italy **1** (2008)

[3] V. Garonne, A.Y. Tsaregorodtsev, I. Stokes-Rees (2004)

[4] F. Berghaus et al., Comput. Softw. Big Sci. **4**, 4 (2020)

[5] B. Holzman et al., Comput. Softw. Big Sci. **1**, 1 (2017), `1710.00100`

[6] J. Balcas et al., J. Phys. Conf. Ser. **664**, 062030 (2015)

[7] E. Kühn, *Online analysis of dynamic streaming data* (2018)

[8] 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`

[9] 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`

[10] 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)
[11] M. Fischer, E. Kuehn, M. Giffels, M.J. Schnepf, A. Petzold, A. Heiss, EPJ Web of Conferences **245**, 07040 (2020)
[12] Kuehn, Eileen, Fischer, Max, Lange, Sven, Petzold, Andreas, Heiss, Andreas, EPJ Web Conf. **245**, 07039 (2020)
[13] *Singularity*, https://sylabs.io, accessed on 2021-02-08
[14] *Charliecloud*, https://hpc.github.io/charliecloud, accessed on 2021-02-08
[15] *HEP_OSlibs meta-package*, https://gitlab.cern.ch/linuxsupport/rpms/HEP_OSlibs/blob/master/README.md, accessed on 2021-02-11
[16] *Universal Middleware Distribution Workernode meta-package*, https://twiki.cern.ch/twiki/bin/view/LCG/EL7WNMiddleware#Description, accessed on 2021-02-11
[17] *CVMFS*, https://cernvm.cern.ch/portal/filesystem, accessed on 2021-02-08
[18] *cvmfsexec: Mount cvmfs repositories as an unprivileged user.*, https://github.com/cvmfs/cvmfsexec, accessed on 2021-02-08
[19] D. Dykstra, J. Phys. Conf. Ser. **331**, 042008 (2011)
[20] *Foreman*, https://theforeman.org, accessed on 2021-02-08
[21] *Puppet*, https://puppet.com, accessed on 2021-02-08
[22] P. Wienemann, O. Freyermuth, *Puppet module for COBalD/TARDIS based opportunistic resource management*, https://github.com/unibonn/puppet-cobald, accessed on 2021-02-08
[23] *Docker Hub*, https://hub.docker.com, accessed on 2021-02-08
[24] O. Freyermuth, P. Wienemann, P. Bechtle, K. Desch, Computing and Software for Big Science **5**, 9 (2021)
[25] R. Caspart, M. Fischer, M. Giffels, R.F. von Cube, C. Heidecker, E. Kuehn, G. Quast, A. Heiss, A. Petzold, EPJ Web Conf. **245**, 07007 (2020)
[26] *The research high performance computer ForHLR II*, https://www.scc.kit.edu/en/services/10398.php, accessed on 2021-02-11
[27] *condor-git-config: dynamically configure an HTCondor node from a git repository*, https://github.com/maxfischer2781/condor-git-config, accessed on 2021-02-11
[28] *KIT Procures New Supercomputer*, https://www.scc.kit.edu/en/services/horeka.php, accessed on 2021-02-22
[29] *LRZ Compute Cloud Service*, https://doku.lrz.de/display/PUBLIC/Compute+Cloud, accessed on 2021-02-11
[30] *Open Stack*, https://www.openstack.org/, accessed on 2021-02-11
[31] *ATLAS Grafana Site-oriented Dashboard*, https://monit-grafana.cern.ch/goto/Et2EdcyGz, accessed on 2021-02-23
[32] *Telegraf*, https://www.influxdata.com/time-series-platform/telegraf/, accessed on 2021-02-23
[33] *Prometheus*, https://prometheus.io, accessed on 2021-02-23
[34] *Elasticsearch*, https://www.elastic.co, accessed on 2021-02-23
[35] *Apel accounting*, https://wiki.egi.eu/wiki/APEL, accessed on 2021-02-23