CRIC: Computing Resource Information Catalogue as a unified topology system for a large scale, heterogeneous and dynamic computing infrastructure

Alexey Anisenkov[1,2,*], *Julia* Andreeva[3], *Alessandro* Di Girolamo[3], *Panos* Paparrigopoulos[3], and *Boris* Vasilev[3]

[1]Budker Institute of Nuclear Physics, Novosibirsk, Russia
[2]Novosibirsk State University, Novosibirsk, Russia
[3]CERN, Geneva, Switzerland

Abstract. CRIC is a high-level information system which provides flexible, reliable and complete topology and configuration description for a large scale distributed heterogeneous computing infrastructure. CRIC aims to facilitate distributed computing operations for the LHC experiments and consolidate WLCG topology information. It aggregates information coming from various low-level information sources and complements topology description with experiment-specific data structures and settings required by the LHC VOs in order to exploit computing resources.
Being an experiment-oriented but still experiment-independent information middleware, CRIC offers a generic solution, in the form of a suitable framework with appropriate interfaces implemented, which can be successfully applied on the global WLCG level or at the level of a particular LHC experiment. For example there are CRIC instances for CMS[11] and ATLAS[10]. CRIC can even be used for a special task. For example, a dedicated CRIC instance has been built to support transfer tests performed by DOMA Third Party Copy working group. Moreover, extensibility and flexibility of the system allow CRIC to follow technology evolution and easily implement concepts required to describe new types of computing and storage resources.
The contribution describes the overall CRIC architecture, the plug-in based implementation of the CRIC components as well as recent developments and future plans.

1 Introduction

The Worldwide LHC Computing Grid (WLCG) [1] is a global collaboration of the computing centers, in more than 40 countries, with the main goal to provide a resource to store, distribute, process and analyze data generated by the Large Hadron Collider (LHC).

More than ten thousand resources, that participate in the LHC experiments independently of the location of their institution, should have transparent access to the LHC data and necessary resources to perform their analysis. The majority of the institutions participating in the LHC scientific program own large computing resources. Therefore, the computing system for LHC was designed by integrating resources of the distributed computing centers in a

[*]Alexey.Anisenkov@cern.ch

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).

single LHC computing service. This design was based on the concept of the computing grid. WLCG is the world's largest computing grid relying on several GRID infrastructures – the EGI [2] and NorduGrid [12] infrastructures in Europe and Open Science Grid in the US[3]. WLCG successfully supported LHC scientific program for almost 10 years.

The next generation of the LHC experiments - High Luminosity LHC (HL LHC) brings new computing challenges. The system should be able to manage multi-Exabyte of data per year and would require about 20M cores for data processing. In contrast with the first years of LHC data taking, when WLCG hardware resources were more homogeneous, the infrastructure should be able to integrate very heterogeneous resources including HPC, specialized clusters, commercial and non-commercial clouds. Moreover, it implies usage of different types of computing platforms like CPU, GPU and FPGAs. A more or less static computing infrastructure is moving towards dynamic one, where elasticity comes in place of static capacity. All those new trends have a strong impact on the WLCG Information System. The primary goal of the information system is to describe all kind of distributed resources used for the computing activities of the LHC experiments. Description of the dynamic and heterogeneous distributed computing environment is a complex task.

2 Main concept of the CRIC system

Computing Resource Information Catalogue, as a high-level information system, is focused onto describe the topology of the WLCG infrastructure as well as experiment-specific configuration required to exploit this infrastructure according to the experiments computing models.

As the evolution of the ATLAS Grid Information System (AGIS) [4], CRIC inherits from AGIS the main concept of resource declaration. It aims for clear separation of the description of the resources provided by the distributed sites vs information which is required to use these resources by the experiments.

CRIC isn't tightly coupled to the WLCG, even thought it's developed and maintained by the WLCG operations coordination team. It can be easily deployed and configured by any collaboration, even beyond LHC. The agile model of CRIC deployment allows anyone to spawn a Core CRIC instance with all the main functionality coming out of the box (computing and storage models, authorisation, authentication, logging, interfaces, APIs etc). This, in addition to the modular and easily extendable architecture, enables any team or experiment to build on top of CRIC, describing their own topology. This description of their resources can then be used to configure any possible third party clients, for example, CRIC can be used to easily configure the Rucio data management framework [5].

3 Architecture

The design of CRIC has been driven by the experience gained so far in operating the ATLAS Distributed Computing environment using AGIS, as well as by the evolution of the WLCG world and the challenges of High Luminocity LHC. Analysis of the infrastructure's topology and configuration data flows as well as collaboration with various communities inside and outside the LHC scope allowed the CRIC team to understand synergies and to properly construct CRIC data models.

CRIC is following a modular architecture based on the Django framework [6]. Modularity is achieved by defining clear building blocks that can be combined to create different CRIC instances. Every building block can be modified by the needs of each client and plugged back in to create a personalised CRIC deployment.

"Provided by" type of information is described in the core part of CRIC, which is generic and serves all experiments. "Used by" type of info which is mostly experiment-oriented is

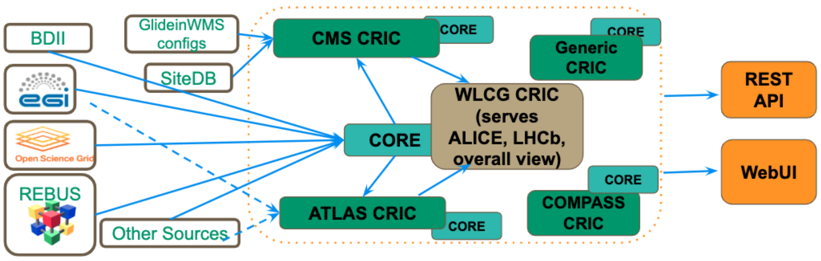

Figure 1. CRIC's architecture

implemented in the experiment-specific plugins. Such a structure allows customizing functionality required by various experiments in the experiment plugins while performing common tasks, like collecting data from the primary information sources, in CRIC core.

Data collectors fetch, analyse and transform information from any source to feed it to CRIC. The architecture of CRIC's data collectors is modular. Anyone can create their own collector which can follow any logic on how data should be transformed and stored into CRIC. These collectors can run in a scheduled manner (using predefined time intervals) or, if needed, they can also run on demand.

Many modern tools and technologies (such as Bootstrap, jQuery and Web services) are used for the development of the user interfaces, while for applications, data is exposed via REST APIs which are configurable by filters and different views. The overall architecture is summarized in Figure 1.

Flexibility and extensibility of the system are essential to enable the description of very dynamic and heterogeneous resources and to follow the technology evolution. The goal is to provide a modular implementation that allows us to be able to re-use the various components, to extend existing functionality or introduce a new one.

4 Deployment model

CRIC consists of different plugins which are utilising the shared building blocks. The core plugin hosts all the client-agnostic information. This includes all the topology models (without their experiment-specific configuration), authentication, authorization, logging, base UI implementations, base API implementation etc.

Core CRIC plugin can be deployed on it's own using the newly developed Docker [7] containers. This can give communities that want to try CRIC an out of the box solution for their use cases. Docker container allows the deployment to any kind of infrastructure without tightening the process to any unnecessary dependencies.

LHC experiments use their own dedicated CRIC plugins. These plugins can be deployed alongside with the core plugin and they include the full description of the experiment computing and storage configuration. Currently experiment specific plugins have been developed for ATLAS and CMS. For LHCb and ALICE, which currently intend to rely on their internal systems for 'used-by' type of information, some limited set of experiment-specific configuration (only the part which is required for central operations) is provided in the WLCG CRIC instance.

In addition to the experiment-specific plugins there are other general interesting plugins developed. The biggest of them is the WLCG plugin which enables tasks required for central operations. Among those tasks is managing pledge information, taking over from the REBUS system [8], generating accounting reports for the WLCG configuration and offering interfaces for accounting data validation by site admins. Another general interest plugin is the DOMA plugin witch includes the interfaces that are used to configure Rucio using data from CRIC.

ATLAS and CMS CRIC instances can work autonomously without any dependency on the WLCG CRIC instance. However, in order to minimize operational effort for validation of information coming from various external information sources, ATLAS and CMS CRIC instances will use WLCG CRIC as a primary information source for their core CRIC components. In this scenario, all external data collectors related to core CRIC data structures will be enabled only for the WLCG CRIC instance. There, data will be centrally checked and modified if required and then propagated to the ATLAS and CMS instances.

5 Status overview

5.1 CMS CRIC

CMS CRIC is in production since 2018. It has already substituted the old CMS topology system (SiteDB) and it's now the main informational source for CMS computing. All the user, group, roles and privileges used by CMS services are configured in CRIC and provided to all the relevant services from there. As of this year CRIC is also used to streamline the configuration management of GlideinWMS[9] Factories for CMS Support. GlideinWMS is a workload management and provisioning system that allows sharing computing resources distributed over independent sites. Up to now the configuration of the system was done by manually editing XML files, uploading them to test servers to verify that the system is not breaking and then proceed to upload them to production. CRIC is now used to to automatically generate the GlideinWMS factory configurations using a modern and safe UI and it's powerful APIs. CRIC's future plans for CMS include the continuation of support of the experiment and the assistant of the current migration of CMS's data management system from Phedex to Rucio. This work is performed in close collaboration with the CMS computing community.

5.2 ATLAS CRIC

Design of CRIC has been inspired by the experience gained in operating ATLAS Distributed Computing environment using AGIS. Since CRIC is now a production ready service, this year ATLAS has started gradual migration from AGIS to CRIC. It is foreseen that AGIS will be completely migrated to the dedicated ATLAS CRIC instance which will become the main topology system for the experiment. In order to guarantee a seamless migration, backwards compatible APIs have been implemented. This way it is ensured that no significant changes would be required for ATLAS clients. CRIC will continue to sync data from AGIS and advertise it's own APIs during the migration period. To make migration transparent for the user community, AGIS will still be the master for the information even when "edit" and "create" functionalities migrate to CRIC. Behind the scene data will be synchronized between two sources. Once all the functionality is enabled in CRIC data creation and modification interfaces will be disabled in AGIS and all necessary requests will be redirected to CRIC. The basic topology, blacklisting APIs, Panda[15] software releases and DDMEndpoints have been already migrated to CRIC. Functionality related to managing Panda queues and Rucio Storage Elements is foreseen to be migrated soon.

5.3 WLCG CRIC

WLCG CRIC has been in production since September 2018. WLCG CRIC is the central CRIC instance serving the needs of the WLCG central operations. It provides a complete description of the topology and generic configuration of the WLCG resources used by all four LHC experiments. WLCG CRIC is being actively used by the WLCG Operations Coordination team, for example in the massive campaign of storage upgrade which is ongoing on the WLCG infrastructure. Some new functionality enabled in the WLCG CRIC is described more in detail below.

5.4 Accounting data validation

Accounting information is one of the important indicators which defines the quality of service provided by a particular site which is a part of WLCG. The amount of storage and compute resources provided by WLCG centers is defined in, so called, pledge values. APEL[13] is an accounting system that collects accounting data from sites participating in WLCG and other GRID infrastructures. This accounting information is gathered from different sensors into a central accounting database where it is processed to generate statistical summaries that are available through the EGI/WLCG Accounting Portal. WLCG Operations recently developed similar system for storage space accounting information - WLCG Storage Space Accounting (WSSA)[14]. APEL and WSSA accounting metrics are being collected by WLCG Accounting Utility (WAU) in order to provide a complete view of CPU and storage accounting information. One of the problems of the current accounting data flow is that sometime metrics generated by APEL or WSSA are wrong due to possible misconfiguration of the local sensors, site batch or storage systems. That is why a procedure which provides a possibility for site administrators to check monthly accounting information, crosscheck it with local site accounting metrics and correct it in case of spotted inconsistencies has been put in place. WLCG CRIC plays there an important role. A CRIC collector, which retrieves accounting summaries from WAU, has been developed together with a set of interfaces which allow to view and validate these data. The process of enabling this functionality was quite straight forward since it used advanced CRIC authentication authorization, table view, forms and other CRIC building blocks. After validation, validated metrics are pushed into WAU, so that it can keep track of both raw and validated data as well as the difference between the two in order to follow up on eventual problems with APEL and WSSA.

5.5 Generation of monthly accounting reports

Another important functionality which has been enabled in WLCG CRIC is generation of monthly accounting reports. So far this task has been performed by the EGI accounting portal. Taking over this functionality by CRIC resulted in improvement of the accounting data flow following the needs of the WLCG operations. New accounting reports are based on validated accounting data instead of the generated one. This allows to substantially improve data quality of the monthly accounting reports.

5.6 Replacement of REBUS functionality

REBUS is a component of the WLCG Information Infrastructure (WLCG II) which contains pledge information and topology of the WLCG federations. All REBUS functionality has been ported to CRIC. This would allow decreasing the number of components of the WLCG II that need to be supported. The plan for REBUS retirement has been agreed with WLCG Management Board. According to this plan, REBUS will be retired by summer 2020 after thorough validation of the new CRIC functionality.

6 New Features

6.1 Live Table editing

CRIC, being the main topology system of WLCG, is constantly evolving and incorporating more use cases. The different nature of data that CRIC has to store requires different ways of processing them. The need of WLCG Project Office to manage multiple pledges at once has driven the CRIC development team to enable bulk editing functionality. To do that a fully customizable mechanism that allows specific fields to be edited directly on the table views has been implemented. That way users don't have to open multiple forms to edit information related to various object instances. Taking advantage of the base data table implementation, bulk editing functionality became a CRIC-wide feature which works in accordance with defined authorisation rules and data validation patterns.

6.2 Permission request and approval wizard

CRIC authentication and authorisation system allows CRIC admins to define user groups and attach different sets of permissions to them. This, in addition to the fine grade authentication on the level of different objects, creates a powerful user management tool. For example site admins can edit all site level objects of their site while data managers of the same site can edit only storage related services. These authorisation groups can be linked to CERN e-groups or be pre-populated with users. Sometime, there is no reliable source of information for bootstrapping of such groups or certain permissions have to be granted to individual users. For such cases a new permission request and approval wizard has been developed by the CRIC team. Users can request permissions using a simple wizard which notifies by email CRIC administrators. CRIC admins can then validate and approve or reject those requests. The approval or rejection can be partial if the admin believes that some permission should not be granted.

7 Conclusions

Over past years CRIC has evolved towards central topology system for CMS and WLCG. It has been successfully used by many different services as a data source for topology, configuration, user groups and permissions information. Further development of the system is being driven by the evolution of the WLCG infrastructure which becomes more dynamic and heterogeneous.

References

[1] I. Bird, Computing for the Large Hadron Collider, *Annual Review of Nuclear and Particle Science*, **61**, 99-118 (2011)

[2] G. Mathieu, A. Richards, J. Gordon, C.D.C. Novales, P. Colclough, and M. Viljoen. GOCDB, a topology repository for a worldwide grid infrastructure, J. Phys.: Conf. Ser. **219**, 062021 (2010)

[3] OIM: https://oim.opensciencegrid.org

[4] A. Anisenkov, A.D. Girolamo and M.A. Pradillo AGIS: Integration of new technologies used in ATLAS Distributed Computing J. Phys.: Conf. Ser. **898**, 92023 (2017)

[5] Barisits, M., Beermann, T., Berghaus, F. et al. Rucio: Scientific Data Management Computing and Software for Big Science **3**, 11 (2019)

[6] Django Project: https://www.djangoproject.com/

[7] Docker: https://www.docker.com/

[8] REBUS: https://gstat-wlcg.cern.ch/apps/topology/

[9] http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/doc.prd/

[10] The ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider , *Journal of Instrumentation*, **3**, S08003 (2008)

[11] The CMS Collaboration, The CMS Experiment at the CERN LHC, *Journal of Instrumentation*, **3**, S08004 (2008)

[12] Eerola, Paula; et al. (2003). "The NorduGrid production Grid infrastructure, status and plans". Proceedings of Fourth IEEE International Workshop on Grid Computing: 158–165.

[13] APEL, Accounting Processor for Event Logs, http://goc.grid.sinica.edu.tw/gocwiki/ApelHome

[14] J. Andreeva, D. Christidis, A. Di Girolamo, O. Keeble, WLCG space accounting in the SRM-less world, EPJ Web Conf. **214** (2019)

[15] Maeno, T. and De, K. and Wenaus, T. and Nilsson, P. and Stewart, G.A. and Walker, R. and Stradling, A. and Caballero, J. and Potekhin, M. and Smith, D. (2011). "Overview of ATLAS PanDA workload management". J. Phys.: Conf. Ser. **331**, 072024 (2011)