Journal of Physics: Conference Series **664** (2015) 032001 doi:10.1088/1742-6596/664/3/032001

Prototype of a production system for Cherenkov Telescope Array with DIRAC

L Arrabito[1], J Bregeon[1], A Haupt[2,3] for the CTA Consortium
R Graciani Diaz[4], F Stagni[5] A Tsaregorodtsev[6] for the DIRAC Consortium

[1] Laboratoire Univers et Particules, Université de Montpellier II Place Eugène Bataillon - CC 72, CNRS/IN2P3, F-34095 Montpellier, France
[2] Deutsches Elektronen-Synchrotron, Platanenallee 6, 15738 Zeuthen, Germany
[3] Department of Physics and Electrical Engineering, Linnaeus University, 35195 Växjö, Sweden
[4] University of Barcelona, Diagonal 647, ES-08028 Barcelona, Spain
[5] CERN
[6] Centre de Physique des Particules de Marseille, 163 Av de Luminy Case 902, CNRS/IN2P3, 13288 Marseille, France

E-mail: arrabito@in2p3.fr

Abstract. The Cherenkov Telescope Array (CTA) — an array of many tens of Imaging Atmospheric Cherenkov Telescopes deployed on an unprecedented scale — is the next generation instrument in the field of very high energy gamma-ray astronomy. CTA will operate as an open observatory providing data products to the scientific community. An average data stream of about 10 GB/s for about 1000 hours of observation per year, thus producing several PB/year, is expected. Large CPU time is required for data-processing as well for massive Monte Carlo simulations needed for detector calibration purposes. The current CTA computing model is based on a distributed infrastructure for the archive and the data off-line processing. In order to manage the off-line data-processing in a distributed environment, CTA has evaluated the DIRAC (Distributed Infrastructure with Remote Agent Control) system, which is a general framework for the management of tasks over distributed heterogeneous computing environments. In particular, a production system prototype has been developed, based on the two main DIRAC components, i.e. the Workload Management and Data Management Systems. After three years of successful exploitation of this prototype, for simulations and analysis, we proved that DIRAC provides suitable functionalities needed for the CTA data processing. Based on these results, the CTA development plan aims to achieve an operational production system, based on the DIRAC Workload Management System, to be ready for the start of CTA operation phase in 2017-2018. One more important challenge consists of the development of a fully automatized execution of the CTA workflows. For this purpose, we have identified a third DIRAC component, the so-called Transformation System, which offers very interesting functionalities to achieve this automatisation. The Transformation System is a 'data–driven' system, allowing to automatically trigger data-processing and data management operations according to pre-defined scenarios. In this paper, we present a brief summary of the DIRAC evaluation done so far, as well as the future developments planned for the CTA production system. In particular, we will focus on the developments of CTA automatic workflows, based on the Transformation System. As a result, we also propose some design optimizations of the Transformation System, in order to fully support the most complex workflows, envisaged in the CTA processing.

1. Introduction

The CTA [1] production system is responsible for the full data-processing and archive in a coherent and automatized way. In this paper we present the status of the prototype of the CTA production system, based on the DIRAC framework [2][3]. DIRAC has been originally developed to support the production activities of the LHCb [4] (Large Hadron Collider Beauty) experiment and today is extensively used by several particle physics and biology communities. DIRAC offers powerful job submission functionalities and can interface with a palette of heterogeneous resources, such as grid sites, cloud sites, HPC centers, computer clusters and volunteer computing platforms. Moreover, DIRAC provides a layer for interfacing with different types of resources, like computing elements, catalogs or storage systems. Developed in python, DIRAC is composed by a collection of sub-systems, each constituted of services, agents, and a database backend. Examples of sub-systems are the Workload Management System (WMS), the Data Management System, the Transformation System, etc. Each system comprises a generic part, which can be easily extended according to the needs of a specific community. As of today, within CTA there has been no specific need to extend any DIRAC sub-system and the CTA extension is limited to the python API for the job submission and CLIs. However, in view of the future Level 1 data-processing, we have started some development work to improve the DIRAC Transformation System, which will be specifically described in this paper.

The paper is organized as follows. In section 2 we will give an overview of the CTA production system, briefly describing the role of its main components. The deployment and the exploitation of the current prototype of the production system is presented in section 3. Finally, in section 4 we present the DIRAC Transformation System architecture, as well as the current developments and future plans.

2. Overall view of the CTA production system

The CTA data-processing, from raw data to high level data, is composed of a set of complex software pipelines (calibration, reconstruction, analysis, Monte Carlo), each one being composed of several application 'stages'. A sequence of stages executed within a single job corresponds to a 'task'. A collection of identical tasks, with a varying parameter, forms a 'production' (see Fig. 1). The full processing is thus composed of several productions, each one eventually associated to a given dataset.

The CTA production system is based on four main components: 1) Computing Resource Management System (CRMS); 2) Archive System; 3) Pipeline System; 4) Information and Communications (IC) infrastructure. The CRMS manages the whole job's life-cycle, from submission to execution on the IC infrastructure. It is composed of two main sub-systems: the first deals with individual jobs (Workload Management System), while the second intervenes at a higher level, dealing with collection of jobs, i.e. 'productions' in the sense specified above. The Archive System is in charge of all the data management tasks, i.e. data ingestion/retrieval, replication and removal. Additionally, it includes a replica and meta-data catalog. The Pipeline System handles the execution of the CTA workflows at the level of the single job. Finally, the IC infrastructure comprises the computing, storage and network resources.

3. Current prototype of the CTA Production system for Monte Carlo simulations

In 2011, we have started the evaluation of the DIRAC system for the CRMS, and in particular the DIRAC WMS [5]. In this context, we have deployed a first prototype, composed of four servers, hosted at PIC and CC–IN2P3. Servers run all the necessary DIRAC services and agents, host the associated databases, as well as the DIRAC web portal. In addition, as the Archive System was still in early development phase, we have also adopted the DIRAC Data Management System, the DIRAC File Catalog and the LHC File Catalog in order to have a fully working framework for Monte Carlo (MC) simulation production. The DIRAC Request

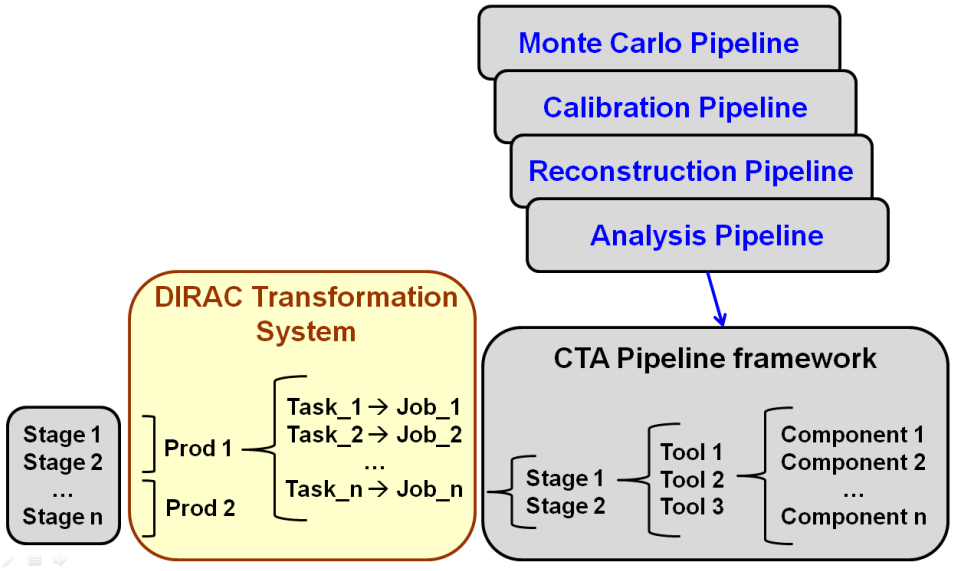

Figure 1. Each pipeline is composed by several stages, tools, components and is built within a common framework (see the bottom right box). A sequence of pipeline stages linked together defines a production, as shown on the bottom left box. Finally, each production is composed by several identical tasks or jobs having a varying input parameter, as for instance the input file to process.

System (RMS) has also been installed to asynchronously treat different types of requests coming from jobs or other DIRAC systems. This is used for instance by jobs to recover failed transfers. The Pipeline System is replaced by simple shell scripts executing applications adapted from the currently running Cherenkov experiments. Finally, for the IC infrastructure we exploit the resources made available by several computing centres of the CTA consortium and under the grid EGI (European Grid Infrastructure) framework.

During the past three years, this prototype has been extensively exploited for CTA Monte Carlo simulations and analysis dedicated to the CTA 'site selection', i.e. the selection of the best location to host CTA telescopes. In the first phase, MC productions were submitted directly to the WMS as a set of parametric jobs, each MC production consisting of about 100k jobs and lasting in total about 5 to 6 weeks. In the second phase, the Transformation System has been introduced to handle MC productions in a more efficient way. The production manager defines the production, also called 'transformation', by defining the workflow to be executed within each job and the varying parameter, which in this case is the MC run number. Once the transformation is created, the production manager deals with a single object, rather than hundreds of thousands. Several actions can then be performed on each transformation, such as starting, extending, stopping, resuming, cleaning or archiving. Moreover, monitoring is very much simplified as the transformation status is directly accessible via a python API and the DIRAC web interface. Finally, human errors are much reduced. As an example, once the transformation is created, new tasks are added by 'extending' the transformation, making sure that all tasks have the same reference configuration.

MC simulations are the most simple example of transformations, since they don't have any input file. However, the Transformation System can be used for many others use-cases, which can be grouped in two categories, i.e. data-processing and data-manipulation. The first category concerns tasks to be treated by the WMS (jobs or 'workflow tasks'), like MC simulation or re-

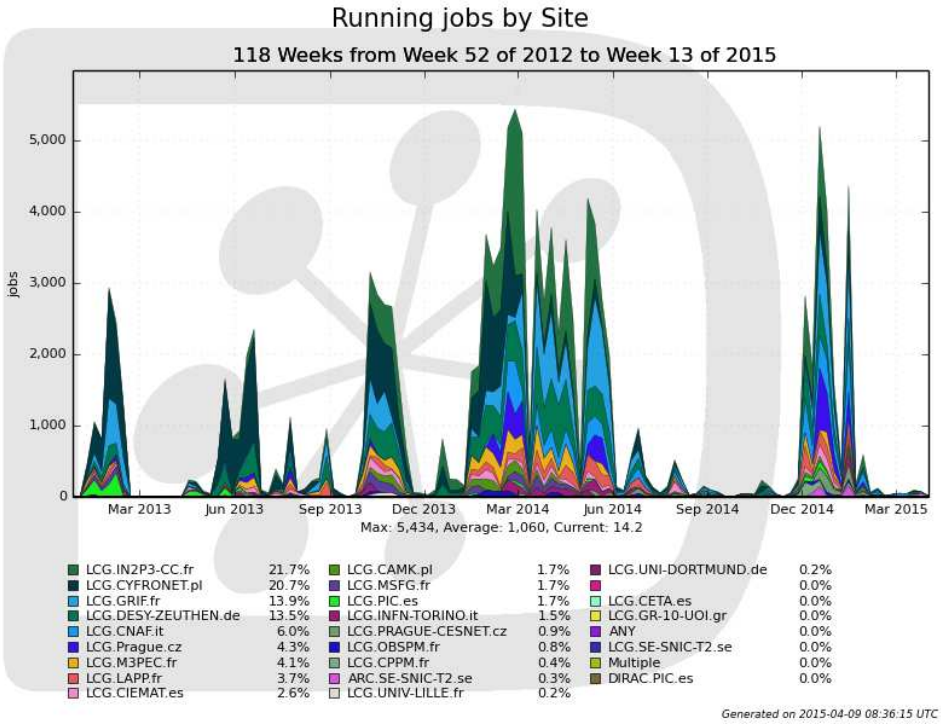

Figure 2. Running jobs from January 2013 until March 2015. Sustained peaks of activtity correspond to the different MC campaigns dedicated to CTA site selection. Job sharing among about 20 grid sites is represented by the colour legend.

processing, while the second concerns tasks to be treated by the RMS ('request tasks'), like for instance bulk data removal, replication or transfer. In the following we will focus more on the first category, since in the CTA production system, data-manipulation will be handled by the Archive System. Besides, transformations can have input files, as in the case of re-processing, or not, as in the case of MC simulations. Input files can be defined either as a fixed list or as the result of a meta-data query. In order to explore the DIRAC Transformation System functionalities, we have implemented a few CTA use-cases, such as re-processing of simulated data, bulk data-removal of obsolete data and data-replication. In 2013-2014, we have run about 10 MC campaigns (see Fig. 2), executing a total of 7 M jobs and consuming 170 M CPU HS06 hours. The deletion of old datasets affected about 10 M of files for a total volume of about 125 TB.

The successful results of the application of the Transformation System to CTA productions, proved that the system has all the basic functionalities needed for the CTA data-processing. Regarding the scalability, we did not perform any dedicated test. However, as we will see in section 4.2, the experience of other users community, like the LHCb experiment, indicates that some improvements can be done in this field.

4. Evolution of the Transformation System: towards an input-data driven system

4.1. Transformation System architecture

As any DIRAC system, the Transformation System architecture follows a service oriented paradigm. Each DIRAC system is composed of databases, services and agents. Databases keep the persistent state of a system and they are accessed by services and agents as a kind

of shared memory. Services are passive components listening to incoming client requests and reacting accordingly by accessing the database back-end. Agents are active components running continuously and invoking periodically their execution methods. They animate the whole system by executing actions, sending requests to DIRAC or third party services. Focusing now on the architecture of the Transformation System, we give below a short description of its main components:

- **TransformationDB**: collects and serves the necessary information to automate the preparation of *tasks* associated to transformations
- **TransformationManager service**: serves the Transformation System clients accessing the TransformationDB
- **Agents**:
  - TransformationAgent: processes the transformations found in the TransformationDB and creates the associated tasks, by connecting input files with tasks according to a given *plugin*
  - InputDataAgent: updates the files to be attached to the active transformations according to a given meta-data query (*inputdataquery*) fetched from the TransformationManager service
  - WorkflowTaskAgent: takes the 'workflow tasks' from the TransformationDB and submits them to the WMS
  - RequestTaskAgent: takes the 'request tasks' from the TransformationDB and submits them to the RMS

Interactions between all these components are shown in Fig. 3. To illustrate the system at work, let us consider the example of a 'data-reprocessing' transformation. The production manager creates a transformation, by defining its attributes. Among these, we mention: the *body*, i.e. the workflow or the request to be executed within a task; the *type* (*e.g.* MCSimulation, data-reprocessing, removal, etc.); the *inputdataquery* and the *plugin*. The *inputdataquery* is the meta-data query which selects the list of files to be associated to the transformation. The way the files are grouped and associated to the transformation tasks is determined by the *plugin*. For instance, files can be grouped according to their physical location, to a given share or until they reach a certain size. Once the transformation is created, it's stored (with its attributes) in the TransformationDB. The InputDataAgent runs continuously querying both the TransformationDB and the Archive catalog in order to select files matching the *inputdataquery*. The selected files are then attached to the corresponding transformations and registered in the TransformationDB. At the same time, the TransformationAgent queries the TransformationDB to determine if there are new transformations to process and creates the corresponding tasks according to a given plugin. Tasks may be jobs (workflow tasks) submitted to the WMS through the WorkflowTaskAgent, or requests submitted to the RMS by the RequestTaskAgent.

In this schema, the Transformation System is data-driven, in the sense that the production manager simply defines a transformation with an *inputdataquery* and the associated tasks are automatically created and executed as soon as new data are registered in the catalog. However, from the previous description, we have seen that internally the Transformation System is not really data-driven, indeed all agents work in 'polling mode' by querying either the TransformationDB or the Archive catalog. The experience of LHCb with the Transformation System, shows that one of the major bottleneck comes from the InputDataAgent continuously performing queries to the Archive catalog. The output of these queries can be indeed very large, resulting in a slowness of the system.

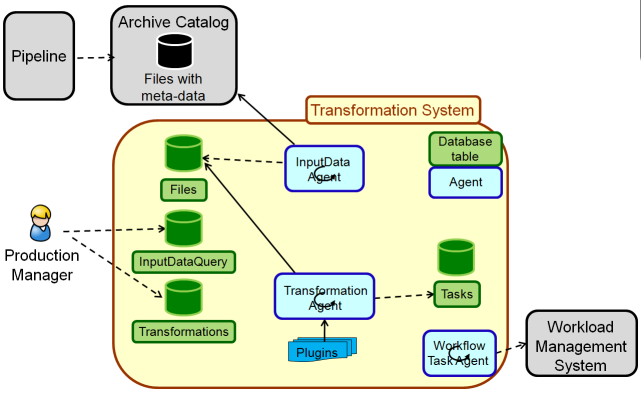

Figure 3. Simplified view of the Transformation System architecture.

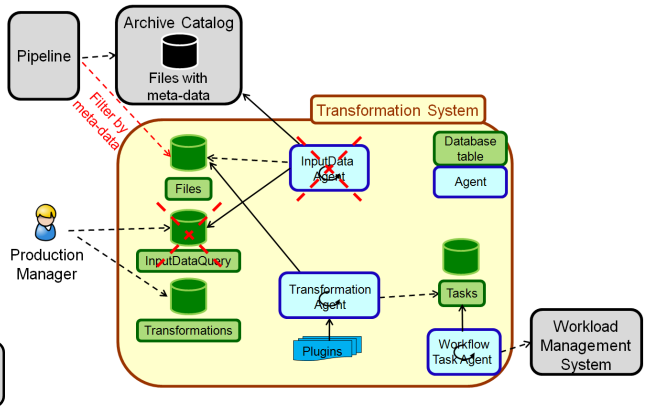

Figure 4. Evolution of the Transformation System architecture. The changes with respect to the current design are marked in red.

4.2. Current and future developments

In order to optimize the performance of the Transformation System, we are currently developing a new version, aiming at a fully data-driven system. The general idea is that when a new file is produced by the Pipeline, it should be immediately filtered against the *inputdataqueries* of the active transformations and automatically attached to the corresponding transformation if needed. To achieve this goal, when the Archive catalog is called to register a new file, the file is always also presented to a meta-filter that checks matching transformations, and is registered in the Transformation System catalog if so. The registration in both catalogs is transparent for the client. This is possible thanks to the fact that the Transformation System was designed to expose a catalog interface and to the fact that it's possible to configure DIRAC to use as many catalogs as needed, provided they expose a standard interface.

Our current developments consist in the implementation of 'meta-filters', which means instrumenting a certain number of methods of the Transformation System catalog with a meta-filter, similarly to what we have just explained for the registration method. In this scenario the InputDataQueryAgent is not needed anymore, however to enforce consistency, it could still work only in background, thus avoiding to stress the Archive catalog and improving the responsiveness of the system. The changes introduced in the new architecture are shown schematically in Fig. 4.

The implementation of meta-filters is the first step to achieve a data-driven system, but it's not enough. Indeed there are other components of the Transformation System working in polling mode, such as the TransformationAgent, the WorkFlowTaskAgent and the RequestTaskAgent. Our plan is to explore the usage of a Message Queuing System to trigger the execution of the three agents mentioned above. In particular, when a new transformation is created in the TransformationDB, a signal would be sent to the TransformationAgent, which prepares the tasks. When a new task is created, another signal would be sent either to the WorkflowTaskAgent either to the RequestTaskAgent so that the task is immediately submitted to the appropriate system (WMS or RMS). The signal can carry some useful information, like for instance which transformations are touched by the event, in order to let the agents perform just well focused operations contrary to their behavior in polling mode. The polling mode would also remain active to recover from the potential failures of the Message Queuing System, like for instance lost signals. This can be reached by associating to each agent two threads: one for the 'Message Queuing mode' listening to the messages, and one for the polling mode listening to the timer.

Finally, in the current Transformation System, each transformation is independent from the

others, so that the different stages of the data-processing, must be defined separately by the production manager. In a completely automatized production system, it should be possible to chain several transformations that are logically linked, as for instance the sequence: re-processing, merging, removal. Our perspectives consist in developing a generalized light system to support chained transformations.

5. Conclusions

A prototype of the CTA production system, based on DIRAC, has been extensively exploited during the past three years, in the context of the MC simulations devoted to the CTA site selection. In particular, the DIRAC Transformation System has been evaluated for the automatisation of the CTA production activities. The successful application of the system to different CTA use-cases, shows that the system has all the basic functionalities needed to handle the future CTA Level 1 data-processing pipeline. Encouraged by these results, we have also started the development of a new version of the system, aiming to improve its performances, through a fully data-driven design. Finally, in order to completely automate the CTA data-processing, our longer term perspectives consist in developing a system able to connect several transformations together.

Acknowledgments

We gratefully acknowledge support from the following agencies and organisations: Ministerio de Ciencia, Tecnología e Innovación Productiva (MinCyT), Comisión Nacional de Energía Atómica (CNEA) and Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) Argentina; State Committee of Science of Armenia; Ministry for Research, CNRS-INSU and CNRS-IN2P3, Irfu-CEA, ANR, France; Max Planck Society, BMBF, DESY, Helmholtz Association, Germany; MIUR, Italy; Netherlands Research School for Astronomy (NOVA), Netherlands Organization for Scientific Research (NWO); Ministry of Science and Higher Education and the National Centre for Research and Development, Poland; MICINN support through the National R+D+I, CDTI funding plans and the CPAN and MultiDark Consolider-Ingenio 2010 programme, Spain; Swedish Research Council, Royal Swedish Academy of Sciences financed, Sweden; Swiss National Science Foundation (SNSF), Switzerland; Leverhulme Trust, Royal Society, Science and Technologies Facilities Council, Durham University, UK; National Science Foundation, Department of Energy, Argonne National Laboratory, University of California, University of Chicago, Iowa State University, Institute for Nuclear and Particle Astrophysics (INPAC-MRPI program), Washington University McDonnell Center for the Space Sciences, USA. The presented work has been partially financed by *Comisión Interministerial de Ciencia y Tecnología (CICYT)* (project FPA2010-22056-C06-02).

References

[1] Actis M et al. (CTA Consortium) 2011 Design concepts for the Cherenkov Telescope Array CTA: an advanced facility for ground-based high-energy gamma-ray astronomy *Experimental Astronomy* **32** 193-316
[2] Casajus A et al. 2012 Status of the DIRAC Project *Journal of Physics: Conference Series* **396** 032107
[3] Tsaregorodtsev A et al. 2014 DIRAC Distributed Computing Services *Journal of Physics: Conference Series* **513** 032096
[4] Stagni F et al. 2012 LHCbDirac: distributed computing in LHCb *Journal of Physics: Conference Series* **396** 032104
[5] Arrabito L et al. 2014 DIRAC framework evaluation for the Fermi-LAT and CTA experiments *Journal of Physics: Conference Series* **513** 032003