AGIS: Evolution of Distributed Computing information system for ATLAS

A Anisenkov¹, A Di Girolamo², M Alandes³, E Karavakis³ on behalf of the ATLAS Collaboration

¹ Budker Institute of Nuclear Physics, Novosibirsk, Russia
² CERN, European Organization for Nuclear Research, Switzerland

Email: Alexey.Anisenkov@cern.ch

Abstract. ATLAS, a particle physics experiment at the Large Hadron Collider at CERN, produces petabytes of data annually through simulation production and tens of petabytes of data per year from the detector itself. The ATLAS computing model embraces the Grid paradigm and a high degree of decentralization of computing resources in order to meet the ATLAS requirements of petabytes scale data operations. It has been evolved after the first period of LHC data taking (Run-1) in order to cope with new challenges of the upcoming Run-2. In this paper we describe the evolution and recent developments of the ATLAS Grid Information System (AGIS), developed in order to integrate configuration and status information about resources, services and topology of the computing infrastructure used by the ATLAS Distributed Computing applications and services.

1. Introduction

The ATLAS experiment [1] at the Large Hadron Collider has successfully collected billions of physics collision events during the Run-1 period (from 2009 to 2013) and is ready to operate in the upcoming LHC Run-2 (from 2015 to 2018) during which tens of petabytes of data will be produced annually. All these petabyte-scale data will need to be stored, processed and analyzed. The ATLAS data are distributed, processed and analyzed at more than 130 grid and cloud sites across the world, according to the ATLAS computing model [2], which is based on a worldwide Grid computing infrastructure that uses a set of hierarchical tiers. It provides to all members of the ATLAS Collaboration speedy access to all reconstructed data for analysis, and appropriate access to raw data for calibration and alignment activities. ATLAS Distributed Computing (ADC) is capable of running more than 150 000 concurrent jobs simultaneously across the grid.

The variety of the ATLAS Computing Infrastructure requires a central information system to define the topology of computing resources and to store the different parameters and configuration data which are needed by the various ATLAS software components. A centralized information system helps to solve the inconsistencies between configuration data stored in the various ATLAS services and in the different external databases. AGIS is the system designed to integrate configuration and status information about resources, services and topology of the computing infrastructure used by ATLAS Distributed Computing applications and services.

Being an intermediate middleware system between clients and external information sources like central gLite BDII (Berkeley Database Information Index) [3], Grid Operations Centre Database (GOCDB) [4], the Open Science Grid Information services (MyOSG, OSG Information Management
System - OIM) [5], AGIS defines the relations between experiment specific used resources and physical distributed computing capabilities.

Being in production during LHC Run-1, AGIS became the central information system for Distributed Computing in ATLAS and it is continuously evolving to fulfill new user requests, enable enhanced operations and follow the extension of the ATLAS Computing model.

In this paper we describe the major capabilities of AGIS as the result of system evolution, that can be expressed in a nutshell as in the following list of actions shown in Figure 1: define, connect, collect, integrate, declare, complete, operate, distribute and finally play – full support for the ATLAS experiment.

2. System capabilities, information and data sources
The ADC applications and services require the diversity of common information, configurations, parameters and quasi-static data originally stored in different sources. Sometimes such static data are simply hardcoded in application programs or spread over different configuration files.

The difficulty faced by the ADC applications is that ATLAS computing uses a variety of Grid infrastructures (European Grid Infrastructure (EGI) [6], Open Science Grid (OSG) [7] and NorduGrid, ARC-based infrastructure [8]) which have different information services, application interfaces, communication systems and policies. Therefore, each application has to know about the proper information source, data structure formats and application interfaces, security infrastructures and other low-level technical details to retrieve specific data. Moreover, an application has to implement a communication logic to retrieve data from external sources that produce a lot of code and effort duplication. AGIS defines the topology of Distributed Computing resources and masks the heterogeneity of computing infrastructures by providing a consistent Computing model definition for application services and developers.

Another point of difficulty in the data organization is the missing links between experiment-specific consumed resources and physical Grid capabilities: the data structures and hierarchy of resources defined in external information services do not fit well in the ATLAS computing relations. For example, ATLAS can define a few sub-sites behind one physical Grid resource. The primary goal of AGIS is to facilitate, enable and define those missing relationships between the physical computing resource provided by various sites and the ones used by the experiment. By providing an abstraction layer from the physical resources, AGIS allows the Experiment to define their own real organization of the resources. This is what we call the connect capability of AGIS.

![Figure 1. Key capabilities of the system.](image-url)
The following two capabilities, namely collect and integrate, underline the features of the system to prepopulate the database content and keep it updated with both static and dynamic information retrieved from external information sources. The system automatically collects topology relations, site specific information required by ATLAS, caches and keeps it up to date, removing the external source as a direct dependency for clients but without duplicating the source information itself. The integrate action is mainly related to dynamic properties like site status, resource state, downtimes, DDM [9] / PanDA [10] blacklisting, PanDA queue dynamic properties and others.

In addition to the collect and integrate functionalities, AGIS also declares within the system various site configuration structures related to the experiment usage of its resources, i.e. Squid Configuration, Frontier Configuration, PerfSonar Configuration, DDM Access Protocols.

The collected information is stored and exposed in a more convenient way to the ATLAS experiment. Supplementary data models and object relations are introduced in the system to cover any experiment specific use-cases and simplify user operations. AGIS completes, organizes, stores and generalizes the information model for the ADC applications and services.

Finally, AGIS provides functionalities to operate information via the User oriented Web Interface [11] portal and to distribute data through unified interfaces (REST style API and Web User Interface).

Figure 2. Experiment view of physical resources.
3. System architecture
The AGIS architecture is based on the classic client-server computing model. AGIS uses the Django framework as a high-level web application framework written in Python. The Oracle DBMS is used as the default database backend.

The object relation mapping technique which is built in the Django framework allows access to the content of the database in terms of high level models, thus avoiding any direct dependence on the relational database system used. Since the system provides various client interfaces such as an Application Programming Interface (API), a Web user interface (WebUI) and a Command Line Interface (CLI) to retrieve and manage the needed data, no direct access to the database from the clients is required.

Figure 3 shows a schematic view of the AGIS architecture. To automatically populate the database with the information collected from external information sources, a set of collectors run on the main AGIS server. All interactions with the external information services are hidden. Synchronization of the AGIS content with the related sources is performed by agents that periodically communicate with sources via standard interfaces and update the AGIS database content. For some types of information AGIS itself is the primary source. The clients are able to update information stored in AGIS through the API and the WebUI. The WebUI is mainly used to define new objects, modify existing properties and easily browse experiment specific resources from various user-friendly views. The AJAX (Asynchronous JavaScript and XML) technology is actively involved to offer efficient interactive access through the WebUI. REST (Representational State Transfer) style API and command line tools further help the end users and developers to use the system conveniently.

From the point of view of data synchronization, the whole information stored in AGIS can be classified as static and dynamic. Dynamic data means regular synchronization against the information source from which it is collected. Technically it could be as a set of dynamic properties (for example, CE queue’s status, free space at a site) or as complete objects (e.g. downtime entries). Objects automatically injected into the system (e.g. GOCDB/OIM sites and services) are registered in the database in DISABLED states (hidden for data export through API), to make such object visible the user has to activate it via the WebUI forms.

Today, the REST full client API allows users to retrieve data in JSON and, for some applications, in XML formats. For instance, all the ATLAS topology can be exported either in the XML format or in JSON structures suitable for the ADC applications.

4. Recent developments
The AGIS main concept of distinguishing between 'used by' and 'hosted by' site resources (the define and the connect capabilities explained above) easily allows the transparent declaration of any virtual resource, such as Cloud and High-Performance Computing (HPC), which have recently become widely used by the ATLAS computing. Following the evolution of the ATLAS Computing model, AGIS is able to define a new top Site entry (see Fig.2, same level as for GOCDB or OIM) related to HPC or Cloud resources, and then all the remaining object definition of computing resources remains the same as for regular grid resources. A special resource_type attribute on the level of PandaQueue object makes it possible for the PanDA system to identify non-grid resources and interpret them appropriately.
The evolution of the ATLAS computing model and its continuous extension required immediate schema updates and implementation of new functionalities on the AGIS side. An example of recently implemented new types of services in AGIS is the support of declaration of HTCondor-CE [14] computing elements which use the next-generation gateway software for OSG sites. AGIS collectors have been upgraded to retrieve information about HTCondor-CE entries defined in BDII, as well as the corresponding WebUI views have been updated to let site administrators also manage HTCondor-CE service definition within AGIS directly.

Figure 4 illustrates another AGIS schema update related to the consolidation of computing resources definition for the ATLAS workload management system. The key concepts of the consolidation consist of preventing data duplication by introducing a parent template object, removing redundant many-to-many PandaResource-PandaQueue relations (historically defined to associate many CEs to the same PandaResource), and simplifying operations in the end. Template based PandaQueue definition allows the inheritance of schedconfig parameters and helps in the consistent declaration of multicore, analysis and production resources behind a PandaSite. Any schedconfig parameters can be shared or overwritten by a child PandaQueue object. Merging objects into a single entry also benefits operation with SWRelease tags more effectively. Moreover, the new implementation provides a functional way to resolve default PandaQueue instance for a given PandaSite, in particular, it helped to incorporate the mapping of high memory and multicore resources required by HammerCloud [15]. The implemented consolidation became the first step in the evolution of computing resource definition in AGIS. The final goal, which is currently under development, is to implement a completely dynamic computing resource definition for PanDA.

Many other functional updates of the WebUI and the API have been implemented and delivered into the production to enhance the user operations and to improve the data management activity. In particular, it includes bulk support of regular operations on DMMEndpoint and PandaQueue objects in the WebUI, as well as the development of new REST style API to apply bulk operations programmatically.

Based on the new REST style bulk API for DDM objects, an automated blacklisting service for DDMEndpoint storage resources has been developed and moved into the production. In its current implementation, the DDM blacklisting service takes into account site downtime information and...
available storage space data (physical free space or user quota limitation) and translates it into special metrics (namely AGIS, DISKSPACE and QUOTASPACE accordingly) to be submitted into AGIS via the REST API. ADC experts and shifters also use the blacklisting API to manually enable or disable space tokens through AGIS.

**Figure 4.** Template-based PandaQueue objects definition. Coherent descriptions of Production, Analysis, High memory and Multi Core PanDA queues.

5. Conclusions
AGIS has been developed to provide, in a single portal, the topology and resources information to configure the ATLAS computing applications. Being in production during the LHC Run 1, AGIS became the central information system for Distributed Computing in ATLAS and represents the primary source of information for all the ADC applications and services.

AGIS functionalities allow the ADC community, experts and shifters to configure and operate production ADC systems and Grid applications. AGIS is continuously extending data structures to follow the new requirements and needs of the ADC, fulfill new user requests, enable enhanced operations and follow the extension of the ATLAS Computing model.

The AGIS design and basic principles included in the architecture allow the use of the AGIS core part by several experiments. AGIS is evolving towards a common information system not coupled to a specific experiment. As a result, CMS is currently evaluating AGIS to use it as the primary information source throughout the experiment to describe its topology (sites and services both SEs and CEs).

AGIS can be used within CMS to map the CMS network topology (PhEDEx nodes to CMS sites to SE names to gridFTP servers), store perfSONAR definitions, store user to physics group associations and quotas on groups, store information about disk space at sites and to manage the configuration of Condor glidein factories.

References
[1] The ATLAS Collaboration 2008 The ATLAS Experiment at the CERN Large Hadron Collider *JINST* **3** S08003
[2] Jones R and Barberis D 2008 The ATLAS Computing Model *J. Phys.: Conf. Ser.* **119** 072020
[3] Grid Information system, http://gridinfo.web.cern.ch
[4] Grid Configuration Database (GOCDB), https://wiki.egi.eu/wiki/GOCDB
[5] Open Science Grid Information service (MyOSG), http://myosg.grid.iu.edu/about
[6] European Grid Infrastructure (EGI), http://www.egi.eu/
[7] Open Science Grid (OSG), http://www.opensciencegrid.org/
[8] NorduGrid project, http://www.nordugrid.org/
[9] Branco M on behalf of the ATLAS Collaboration. 2008 Managing ATLAS data on a petabyte-scale with DQ2 J. Phys.: Conf. Ser. 119 062017
[10] Maeno T on behalf of the ATLAS Collaboration. 2008 PanDA: Distributed production and distributed analysis system for ATLAS J. Phys.: Conf. Ser. 119 062036
[11] AGIS WebUI portal, http://atlas-agis.cern.ch
[12] Django project, http://www.djangoproject.com
[13] REST interfaces, http://en.wikipedia.org/wiki/Representational_state_transfer
[14] HTCondor-CE documentation page, https://twiki.opensciencegrid.org/bin/view/Documentation/Release3/HTCondorCEOverview
[15] Elmsheuser J on behalf of the ATLAS Collaboration. 2014 Grid site testing for ATLAS with HammerCloud J. Phys.: Conf. Ser. 513 032030