GOCDB4, a New Architecture for the European Grid Infrastructure

Gilles Mathieu, John Casson

Gilles.Mathieu@stfc.ac.uk, John.Casson@stfc.ac.uk

STFC, Rutherford Appleton Laboratory, UK

Abstract The tool known as GOCDB, or Grid Operations Centre Data Base, consists of a central authoritative database which contains static Grid resource and topology related data. It stores information about regions, countries, sites, nodes, services and users, and links this information together in a logical way. Within the past years, GOCDB has imposed itself as a central authoritative repository for topology and site information within EGEE and WLCG. As for all other operational tools, the dramatic evolution of EGEE in order to prepare for a sustainable European Grid Infrastructure imposed many changes on GOCDB architecture. One of these changes is the requirement for a distributed architecture, where a central system can collect and display information maintained by regional instances of the system: in May 2010, GOCDB will become the official central topology repository for the European Grid Initiative (EGI), and will propose a regionalised model that will allow National Grids (NGI) to run their own instance of the system while keeping synchronised with the central EGI repository. These new requirements along with the limitations of GOCDB old model (known as GOCDB-3) led us to adopt an innovative database design based on a pseudo object database model (PROM). In this model, constraints and relations are built using meta-data. This allows for a large flexibility in the database schema, thus enabling different instances of the same tool to store different schemas while remaining interoperable. On top of this, a PHP-written input/output module gets and retrieves data in XML format, making the whole system as standard and configurable as possible. After reviewing GOCDB-3 architecture and explaining its limitations and the reasons for changing, the paper will describe GOCDB-4 inner architecture from general system overview down to technical details on database design, application level and standard interfaces. It will show how flexibility is achieved through the use of XML configuration files. Pros and cons of adopted model will also be assessed. The paper will finally review the overall distributed architecture and distribution scenarios, as well as interactions between GOCDB-4 and similar tools.

1. CONTEXT

1.1. EGEE and WLCG

The general context of the work presented here lies within two worldwide grid computing projects: Enabling Grids for E-science (EGEE) [1] and the Worldwide LCG Computing Grid (WLCG) [2]. The EGEE Production Service infrastructure is a large multi-science Grid infrastructure, federating some 250 resource centres world-wide, providing some 40.000 CPUs and several Petabytes of storage.

WLCG is an application-level grid, running 140 sites in 33 countries, some of them a subset of EGEE together with other Grids like Open Science Grid, in order to do the computing for the four experiments running on the Large Hadron Collider at CERN.

1.2. GOCDB purpose and description

The tool known as GOCDB [3] (which stands for Grid Operations Centre Data Base) consists of a central authoritative database which contains static resource and topology related data about EGEE and WLCG. It stores information about regions, countries, sites, nodes, services and users, and links this information together in a logical way.

Information presented in GOCDB is static: data are maintained by the appropriate people with given roles. The reason for these data to be here is the need to have a reference, authoritative list: while dynamic data providers used by Grid middleware – such as BDII [4] or MDS [5] - present lists of resources that are available, GOCDB gives a list of resources that should be available. A short history of GOCDB is presented in [3].

1.3. Related work

1.3.1. OIM

OSG Information Management System (OIM) [6] can be seen as the equivalent of GOCDB for the Open Science Grid [7] Project. It basically fulfils the same func-

tions, storing and providing information about resources, users, services and downtimes. The main difference between OIM and GOCDB lies on the facility provided by OIM to store and map Virtual Organizations (VO). In EGEE this functionality is provided by the Operations Portal, a.k.a CIC portal [8].

1.3.2. HGSM

Hierarchical Grid Site Management (HGSM) [9] is a web-based management tool that stores static information about grid sites, used within the SEE-GRID project [10]. The set of information contained in HGSM is very similar to what GOCDB stores, and HGSM is often labeled as "a light version of GOCDB". Since some SEE-GRID sites are also EGEE sites, there is a clear overlap in terms of functionalities between the 2 tools, resulting in data duplication. Collaboration is under way for HGSM and GOCDB to be fully interfaced, as part of the work described later in this paper.

1.4. Why a new architecture?

The third phase of EGEE, started in May 2008, has brought dramatic changes to the project's operational model compared to EGEE-I and EGEE-II [11]. These changes are proposed in order to achieve a successful transition from a central, project-based model to a sustainable infrastructure built on top of each EGEE region, possibly getting down to country level. This final requirement is a result from the European Grid Initiative (EGI) [12] design study where each participating country has the operational components responsibility of maintaining its own National Grid Infrastructure (NGI). These general evolution ideas have been discussed in EGEE-III OAT strategy document [13].

As a consequence of these evolution requirements, GOCDB had to evolve from the central database with a central interface described in [3] to a distributed model. The resulting architecture should allow for all the following points:

- Regional instances should be able to communicate efficiently with one another.
- In the event of a region not willing to host its data, GOCDB should provide a central "catch all" repository to host this information on their behalf.
- There should be a uniform way of accessing the data across all the regional instances, so that even if distributed, the model could also be seen as one.
- Regional instances should be customizable, i.e. adaptable to local needs.
- An adapter should be provided to allow interoperations with those regions that already have their information repository, without asking them to change their system.

2. GOCDB4 ARCHITECTURE AND DESCRIPTION

2.1. Architecture of a GOCDB4 module

2.1.1. Overview

A GOCDB module comprises or the following components:

- A database, where all data are stored. The system is currently designed to run on an Oracle Database.
- An API which interface the database by providing access routines and low level business logic.
- XML I/O modules which allow data to be sent and received in XML format
- Web services and Programmatic Interfaces, which present data to third party tools and allow input from external systems
- A GUI under the form of a web portal

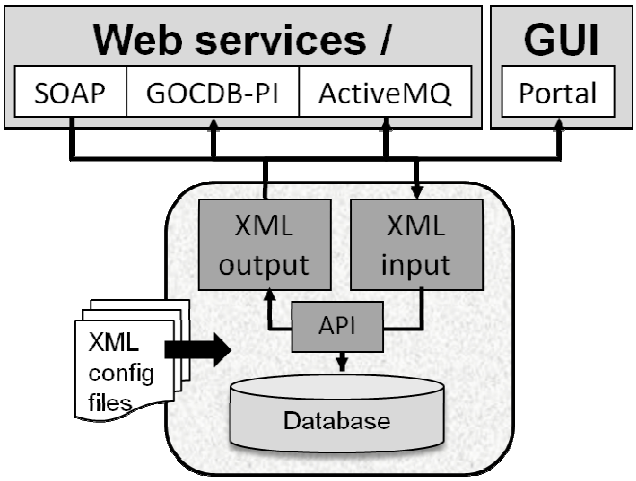

Fig. 1. Overview of a GOCDB module

Most components on top of the database can be configured using XML files, which allows for operating the module without need to change the code at any

time. Fig.1 shows how these components interact with one another. They are described in details in the following sections.

2.1.2. Database model

2.1.2.1. Use of the PROM concept and tools

GOCDB4 database model has been designed and implemented following the pseudo-relational object model (PROM) described in [14]. The main idea behind this design is the concept of removing the physical aspects of any database into effectual 'meta-data'. By doing this it makes it possible to maintain links and table information outside of the actual data tables, allowing this data to be accessed in a uniform way, and changed with minimal, or no, changes to the actual design. This allows for faster deployment, standard data access routines, and the ability to grow the system without the need to redesign or re-implement the actual database.

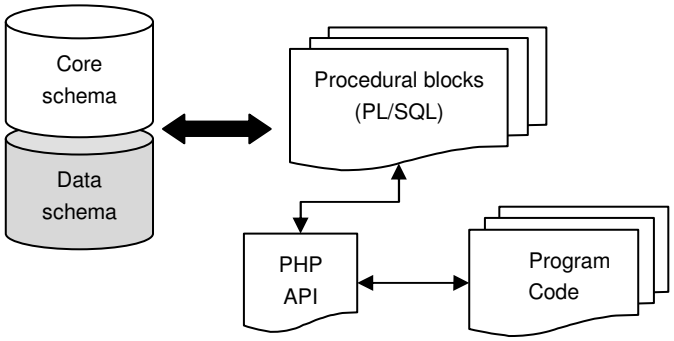

Fig. 2. General schema of a PROM database

The schema described on Fig.2 shows the main architecture of a PROM database as designed by its author. In our GOCDB architecture, the component here described as "program Code" is actually the XML I/O module.

The idea of using a PROM database for GOCDB came out of a design analysis based on the requirements described earlier in this paper. GOCDB previous version, known as GOCDB3, runs on top of a classic relational model as described in [3] and is hosted on an Oracle11g cluster. Distributing this model by splitting it into many instances is not a solution that would allow easy interoperations between these different instances.

The DB design will be required to be a central resource, with the ability to be installed in multiple locations, to cope with multiple grids. It is also understood that each Grid may need or want to vary the design to fit specific needs for that area. PROM seemed a suitable solution to answer these needs because of the inherent flexibility of the model.

2.1.2.2. GOCDB data schema

GOCDB4 data schema is object oriented and is described in [15]. The way PROM works makes it possible to store different objects in the same table provided they have a similar data structure. This is true for many GOCDB objects: Time zones, service types, site statuses are different objects in the PROM way but can all be described by two values: name and description. It is therefore possible to store them in a single database table that will precisely have these fields. This principle then shows a clear differentiation between the *data schema*, which is the description of all object types as well as possible links between them, and the *database schema* which shows which tables are used.

2.1.2.3. Benefits and limitations of the PROM approach

One of the benefits of using a PROM model is its easy deployment: installing and configuring a GOCDB regional instance should be as straight forward as possible. Integrating actual constraints to the dataset allows for such an easy deployment. Moreover, modifications can be made on the logical schema without affecting the overall design which is of great interest considering the tool has to be locally customizable (see 2.1.4.2).

Another benefit comes when thinking of possible future evolutions. At time of writing we don't know exactly how many distributed instance will run in parallel, so building GOCDB new architecture on top of a scalable model is then crucial.

On the side of limitations, performances seem to be somewhat lower than a standard RDBMS, mainly because of the many SQL joins that the model imposes. This is not an issue considering the relatively small size of the database we are dealing with, yet some specific queries have to be cached if we want to guarantee a fast access through the web portal or programmatic interface.

Another limitation to PROM is inherently linked to its flexibility: by not imposing constraints at database level it becomes easier to "break" the schema if PROM principles implemented at application level are not followed. To prevent that from happening data access should only be done through the PROM API and never inserted or modified manually in the database.

2.1.2.4. Adaptations and specific use

One of our key needs for GOCDB4 which is not provided by PROM is the notion of cardinality in object links. From a semantic point of view, some objects can only be linked to 1 object of a given type (e.g. a site to a time zone), while some can be linked to many (e.g. a site to a service endpoint). Even trickier, we needed to define a "level 2" cardinality, e.g. a site can be linked to many groups, but to only one group of each type (one country, one ROC, etc.). This has been achieved by introducing this cardinality in the *gocdb_schema.xml* description file (see 2.1.4.1). This cardinality parameter is then taken into account by the application whenever the need arises to create or modify links between objects.

While low level PROM API provides most of needed routines (see [14]), we found out that one was crucially missing, especially after introducing the notion of cardinality: the ability to simply remove a link between two PROM objects without necessarily deleting one of these objects. The method has consequently been implemented and added to the PROM API, and then interfaced with our code in a similar way to other routines.

2.1.3. Components Description

2.1.3.1. XML Input and Output

Interfacing between database, PROM API and higher level application is done through 2 modules called XML input and XML output. As the name suggest they provide or get data in XML format to/from the database.

XML output creates XML from a database query. A SQL query is read from a configuration file then executed against the database. Each row is then inserted into an XML template element. This template element is repeatedly filled with each row's results and added to the final XML file as described on Fig.3. Parameters can be used and passed to the application when calling the module to create the appropriate query plan.

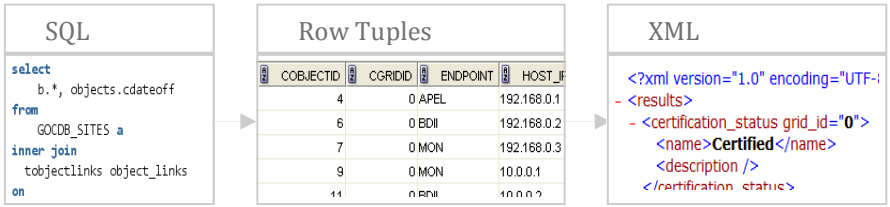

Fig.3. From SQL to XML using the XML output module

XML input works in a similar way with flows in the other direction. It gets XML files, parses them according to predefined templates and calls appropriate PROM routines to insert, edit or delete objects and links in the PROM database.

2.1.3.2. Standard interfaces

GOCDB4 comes with a set of standard interfaces through which data can be gathered or input. The first of those interfaces is a REST [16] interface presenting XML formatted data over https. This interface, known as GOCDB Programmatic Interface or GOCDB-PI, is described in [17]. This interface uses XML templates and SQL queries defined within the XML output module. The GOCDB-PI is mainly used by third party tools that need to collect GOCDB information like site details, services or downtimes.

It is also planned to provide a SOAP [18] interface which will fulfill similar functions, and in addition will be used as the main interface to synchronize the central instance of GOCDB with its regionalized counterparts (see 2.2).

Finally, an Apache ActiveMQ [19] interface is planned in order to interface GOCDB with EGEE messaging system [20]. The main application use case would be to send through ActiveMQ downtime information to the EGEE message bus, thus allowing client application to build calendars or notification systems as appropriate.

2.1.3.3. Web portal

GOCDB4 web portal is a set of pages and scripts that display GOCDB information and allow for modification through a simple web GUI. The code is as application independent as possible, and most of GOCDB specific features are done at configuration level (see 2.1.4).

Initial requirements about the regionalisation of the tool imply that this portal should work on both read-only mode for the central instance and read-write mode for the regional instances (see 2.2). This is also achieved through parameterization in order to keep the code similar between two types of deployment.

GOCDB4 web portal provides a user management system based on X509 certificate DN authentication coupled with role management.

2.1.4. Configuration and Customization

2.1.4.1. Use of configuration files

Most of GOCDB configuration is made through a set of XML files in order to allow for easy changes without the need to touch the code. We've seen above that XML I/O modules were using XML templates to determine the format of the data to send – or to receive. On a more general tone, all GOCDB components work in a similar way in order to keep the code generic.

Initial deployment of a GOCDB module is also facilitated by a schema description file (*gocdb_schema.xml*) which describes in XML format the objects and classes hierarchy, the properties for each class (transformed into database fields) as well as the possible links between objects and their cardinality.

2.1.4.2. Data schema and components customization

PROM's flexibility allows for the data schema to be modified without necessarily breaking the initial version or over-complicating the design, which is a difficult challenge when using relational databases. Since object definition and relations are meta-data, new object types can be added and new link types created without the slightest modification made in the physical database. Customizing GOCDB4 schema in order to adapt to local needs doesn't imply to change the database schema, but only the logical schema.

Equally, it is possible to customize GOCDB interfaces by adding new XML output templates or modifying existing ones with little effort and no coding at all.

2.2. Interactions between modules: The central/regional architecture

As described in 1.4, GOCDB needs to run as a central system relaying information about EGI resources and topology while allowing the actual input system to be distributed and customized. This architecture is described in Fig.4 where three use cases are highlighted:

- Region 1: A region installs and runs its own GOCDB instance. It synchronizes with the central service by pushing its data through standard GOCDB4 interfaces.
- Region 2: A region uses the centrally provided input system
- Region 3: A region uses another tool to operate its topology information and synchronizes with the central GOCDB through an adapter.

The way the PROM model is designed allows for a differentiation of data stored in the same basic structure through the mean of a *collection id* or *grid id*: different objects can be stored in the same table and have different collection ids, thus refer-

ring to a different set of data. This is how data hosted in the central GOCDB system will be "logically split" into different regions. Regionalizing one of these subsets is then a simple question of separating from the main DB all data with the collection ID corresponding to this region.

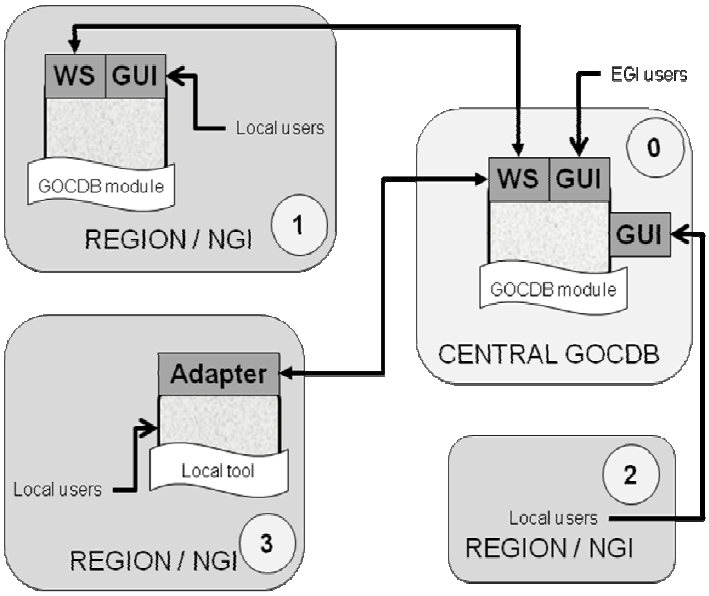

Fig. 4. Architecture of regional and central GOCDB

3. FUTURE WORK AND CONCLUSION

3.2. Future developments

Current Version of GOCB4 is based on Oracle and low level PROM API is provided as an Oracle package of stored functions and procedures. Yet running an Oracle server is a non negligible limitation to GOCDB distribution, mainly because not all National Grids own an Oracle license. There are consequently plans to provide a MySQL version of the system, which will require the low level API to be rewritten.

Another axis for future development is related to the possible integration of GOCDB to other EGI operation tools. This includes the integration of GOCDB and CIC Operations portal [8] under a common interoperable toolkit for grid operations, starting with the integration of a common central human interface allowing users to access both central services through a single entry point, and then providing interoperable back ends for distribution to NGIs as a single package. Such development will require effort at data representation level as well as at interface and data transfer level.

Finally, some work can be done towards standardization of topology data representation. This is the continuation of a design study to better interface dynamic information systems such as the EGEE BDII with static repositories such as GOCDB. This work would require defining a standard schema for storing information (e.g. sites) as well as defining and implementing possible interfacing between static/dynamic models.

3.2. Early analysis of the benefits and downsides of the change

At the time of writing it is difficult to assess the global impact of the change from GOCDB3 to GOCDB4. The new system has to run in production for at least a few months before any useful metrics can be produced. Yet some benefits can already be noticed. Packaging of GOCDB4 module has eased standardization of the code and GOCDB release process. Additionally, the use of a PROM database has already helped coping quickly and easily with schema changes and new requests.

The only obvious downside to the system to be observed at the moment is the relative complexity of the system design: although PROM implements efficient concepts it is sometimes counter intuitive to people used to relational models. Yet experience has shown that once understood, the system is quickly adopted by whoever deploys it and initial reticence should not be considered a showstopper.

REFERENCES

[1] Laure E, Jones R, “Enabling Grids for e-Science: The EGEE Project”. EGEE-PUB-2009-001, *http://cdsweb.cern.ch/record/1128647*

[2] The Worldwide LHC Computing Grid (WLCG), http://lcg.web.cern.ch/LCG

[3] Mathieu G, Richards A, Gordon J, Del Cano Novales C, Colclough P, Viljoen M: “GOCDB, a Topology Repository for a Worldwide Grid Infrastructure”, in *Proceedings of Computing in High Energy and Nuclear Physics (CHEP09), Prague, Czech Republic, March 2009*

[4] Berkeley Database Information Index (BDII),

https://twiki.cern.ch/twiki/bin/view/EGEE/BDII

[5] Czajkowski K, Fitzgerald S, Foster I and Kesselman, C, "Grid information services for distributed resource sharing" in *Proceedings of the 10th IEEE Int. Symposium on High performance distributed computing (HPDC-10), San Francisco, CA, 7–9 August 2001*

[6] OSG Information management System (OIM), *http://oim.grid.iu.edu*

[7] The Open Science Grid Consortium (OSG), *http://www.opensciencegrid.org*

[8] Aidel O, Cavalli A, Cordier H, L'Orphelin C, Mathieu G, Pagano A, Reynaud S, "CIC portal: a Collaborative and Scalable Integration Platform for High Availability Grid Operations", in *Proceedings of The 8th IEEE/ACM International Conference on Grid Computing (Grid 2007), Austin TX, US, Sept. 2007*

[9] Hierarchical Grid Site Management (HGSM), http://hgsm.sourceforge.net

[10] South Eastern European Grid Infrastructure Development (SEE-GRID), *http://www.see-grid.org*

[11] Cordier H, Mathieu G, Schaer F, Novak J, Nyczyk P, Schulz M and Tsai MH, "Grid Operations: the evolution of operational model over the first year", In *Proceedings of Computing in High Energy and Nuclear Physics (CHEP06), Mumbai, India, 2006.*

[12] The European Grid Initiative (EGI) design study, *http://web.eu-egi.eu*

[13] Casey J et al, "Operations Automation Strategy", *https://edms.cern.ch/document/927171*

[14] Colclough P and Mathieu G, "A pseudo object database model and its applications on a highly complex distributed architecture", In *Proceedings of the IARA 1st international conference on advances in databases, knowledge and data applications (DBKDA 2009), Cancun, Mexico, March 2009*

[15] GOCDB4 data schema, *http://goc.grid.sinica.edu.tw/gocwiki/GOCDB4_DB_Schema*

[16] Representational State Transfer (REST) architecture, *http://en.wikipedia.org/wiki/REST*

[17] GOCDB-PI technical documentation, *http://goc.grid.sinica.edu.tw/gocwiki/GOCDB_Technical_Documentation*

[18] SOAP, *http://en.wikipedia.org/wiki/SOAP*

[19] Apache ActiveMQ, *http://activemq.apache.org*

[20] Casey J et al, "MSG - A messaging system for efficient and scalable grid monitoring", *EGEE User Forum, March 09.*