Journal of Physics: Conference Series **331** (2011) 072067 doi:10.1088/1742-6596/331/7/072067

A policy system for Grid Management and Monitoring

Federico Stagni
PH Department, CH-1211 Geneva 23 Switzerland
E-mail: federico.stagni@cern.ch

Roberto Santinelli
IT Department, CH-1211 Geneva 23 Switzerland
E-mail: roberto.santinelli@cern.ch

On behalf of LHCb Collaboration

Abstract.
Organizations using a Grid computing model are faced with non-traditional administrative challenges: the heterogeneous nature of the underlying resources requires professionals acting as Grid Administrators. Members of a Virtual Organization (VO) can use a subset of available resources and services in the grid infrastructure and in an ideal world, the more resoures are exploited the better. In the real world, the less faulty services, the better: experienced Grid administrators apply procedures for adding and removing services, based on their status, as it is reported by an ever-growing set of monitoring tools. When a procedure is agreed and well-exercised, a formal policy could be derived. For this reason, using the DIRAC framework in the LHCb collaboration, we developed a policy system that can enforce management and operational policies, in a VO-specific fashion. A single policy makes an assessment on the status of a subject, relative to one or more monitoring information. Subjects of the policies are monitored entities of an established Grid ontology. The status of a same entity is evaluated against a number of policies, whose results are then combined by a Policy Decision Point. Such results are enforced in a Policy Enforcing Point, which provides plug-ins for actions, like raising alarms, sending notifications, automatic addition and removal of services and resources from the Grid mask. Policy results are shown in the web portal, and site-specific views are provided also. This innovative system provides advantages in terms of procedures automation, information aggregation and problem solving.

1. Introduction

This paper presents an innovative, policy based system for the monitoring, management and usage optimization of Grid resources. We called it Resource Status System (RSS). This paper is organized as follows: section 2 explains in detail the motivations that led to this development and explains how some tools already in use address this use case. Section 3 formalizes section 2, providing a list of requirements. Section 4 explains the development environment while section 5 shows the four steps that compose the RSS cycle, using UML and non-UML diagrams. Conclusions are finally given in section 6.

2. Motivation

Reliable information provided by a solid and mature monitoring system of the Grid infrastructure is crucial for an efficient management of the resources. Grid service suppliers rely on a variety of monitoring tools providing different perspectives on both the infrastructure and activities running from various Virtual Organizations (VOs).

2.1. Information aggregation

The proliferation of monitoring tools brought the VO administrators to interact with a variety of dispersed information somehow difficult to handle successfully. This is one of the main reasons to promote initiatives for aggregating various infrastructure monitoring data, which is a commonly recognized requirement expressed by all VOs. Monitoring information aggregation is realized by a number of tools like GridView, MyOSG, HappyFace and Dashboard ([1]). Dashboard in particular is extensively used by different VOs for different purposes. Dashboard offers a toolkit for monitoring both community activities and status of resources aggregating them in useful views. It is the firm conviction of the authors that these tools represent a very powerful instrument for local resource administrators offering views of multi-VOs activities running on a given site. However a more VO-centric usage is something that requires a lot of integration effort with the experiment specific frameworks in order to fully exploit the information for automation.

2.2. Information combination

Some of the previously presented tools go further: not just information aggregation into a single page but also, using simple algorithms, the possibility to use this information to evaluate the effective health status of a given resource. Such a status is unique and, starting from that, it is easy to dig down into details of the problem. Examples are the Site Status Board (SSB) application from Dashboard or the Availability Computing Engine (ACE) (used by both GridView and Nagios). These tools present however some limitation in terms of:

- target consumers, mostly dedicated to site administrators;
- algorithms used for the combination, very simple and not easily customizable making them not be flexible enough for complex use cases within the HEP collaborations.

2.3. Automation

Once the Grid operations within a team are established with clear procedures in place, running the system becomes a simple and routine job for an experienced Grid administrator. Taking a decision or applying a procedure should be as much as possible an automatic operation. From now on we will refer to such operations as formal policies. As of today either within the infrastructure delivering the grid services or within the operations team of the WLCG [**?**] grid main customers, a tool to evaluate and enforce such policies is missing. The lack of automation in the management of such policies always requires a Grid operations administrator to be available in order to react manually when changes happen in the system.

3. Requirements

The list of motivations expressed in section 2 steered the system requirements hereafter summarized. The system must:

- collect dispersed and uncorrelated monitoring information from all possible and reliable sources available;
- aggregate and expose this information in a simple way;
- sort a global status for each service in a top-down view. It becomes easy to dig into the information presented as soon as one realizes there is a problem somewhere;

- use a policy system to combine information and deliver a decision on top of them. Non-trivial combinations must be possible to take into account the variety of resource natures and the complexity of the systems;
- automate trivial and non-trivial tasks, by enforcing defined policies.

The purpose of the Resource Status System (RSS) is to provide this missing system fulfilling these requirements. The architecture and the functionalities exposed within this paper have been implemented within the LHCb [2] collaboration inside the DIRAC[3] framework.

The RSS evaluates both simple and complex policies and uses them to assess the status of the resources for a given Virtual Organization. Each policy uses information taken from many monitoring sources, each providing a different perspective. Once a status is computed, the RSS can automatically trigger actions accordingly.

4. LHCb and DIRAC

DIRAC (Distributed Infrastructure with Remote Agent Control) [3] is a community Grid solution. DIRAC, which is developed in python, offers powerful job submission functionalities, and a developer-friendly way to create services and agents. A service exposes an XML-RPC implementation; an agent is a stateless light-weight component (comparable to a cron-job).

DIRAC has been initially developed as a LHCb-specific project, but many efforts have been made to re-engineering it into a generic framework, capable to serve the distributed computing needs of a number of Virtual Organizations. After this complex reorganization, the LHCb-specific code resides in the LHCbDIRAC extension while a core, VO-blind, DIRAC project has been disentangled. In this way, other VOs, like Belle II [4], or ILC/LCD [5] have developed their custom extensions to the DIRAC framework.

DIRAC is a collection of sub-systems, each constituted of services. Sub-systems are, for example, the WMS (Workload Management System) or the DMS (Data Management System). Each system comprises a generic part, and a VO-specific part, including the RSS. Since the RSS has been developed within the LHCb VO, the examples that will follow in this paper refer to the way LHCb use it.

5. A RSS cycle

At the moment, the RSS uses up to 7 DIRAC agents. These agents continuously cycle through the list of resources, triggering the information retrieval, the policies evaluation, and eventually the status enforcing. The policy system is the central component of the RSS, and its architecture has been inherited from the well-accepted architecture formerly developed for the security frameworks [6], [7]. We prefer the terminology used for the XACML [8] specification, where a Policy Decision Point (PDP), a Policy Enforcement Point (PEP) and a Policy Information Point (PIP) are specified and used.

The RSS cycle is explained using a logical separation into four steps.

5.1. Step 1: Resources

The resources are organized in a simple ontology which is persisted in a database. Figure 1 explains the ontology used for LHCb, that considers primarily the case of LCG [9] Grid sites.

The UML diagram of figure 2 shows in a simplified way how this step is accomplished within the RSS implementation. First of all, a set of agents is used to get the resources that have to be checked from those persisted in the database. A PEP object is created for each resource selected, which in turns asks a PDP object to evaluate the status of such resource.

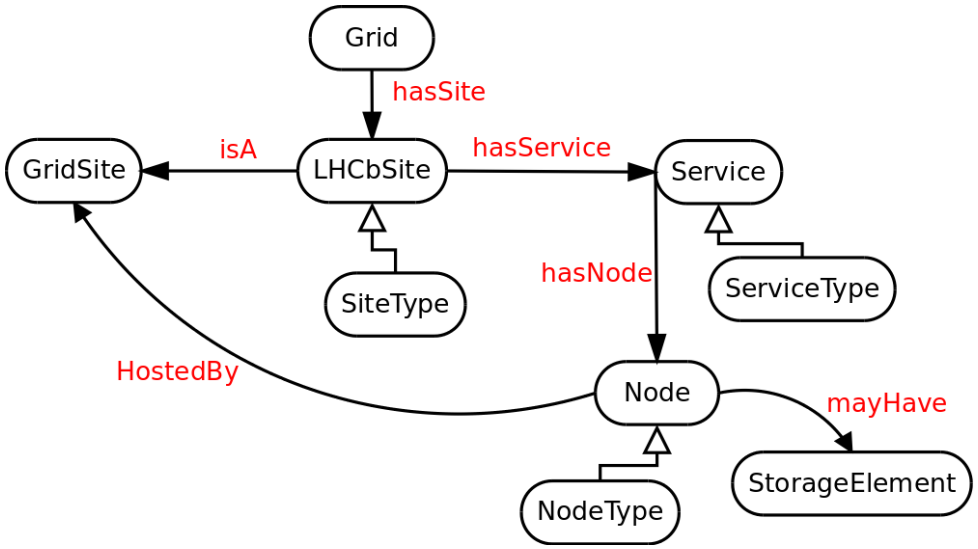

Figure 1. Grid ontology

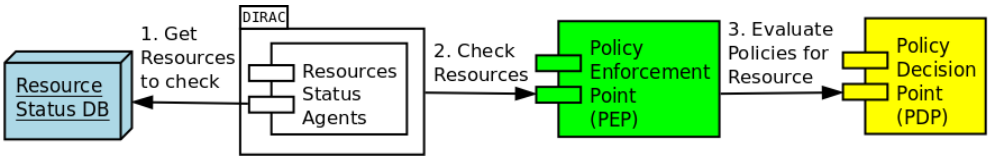

Figure 2. UML diagram of the first step of the cycle

5.2. Step 2: Policies

Figure 3 explains with a simple UML diagram the second step. The PDP contacts the PIP to know which policies have to be evaluated for the selected resource. Once a policy object is created it evaluates a status using monitoring information. Such information is sometimes taken in real time from a remote source, while at other times, in order to reduce the network traffic, an agent asynchronously caches large amounts of data in a local DB.

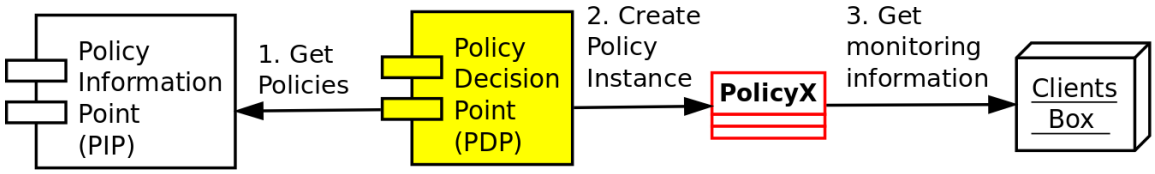

Figure 3. UML diagram of the second step of the cycle

For LHCb, the 4 statuses of figure 4 are used: Active, Bad, Probing, Banned. It has to be noted that these statuses are in principle only internal to the RSS. They can assume a meaning outside the RSS during the enforcing phase.

5.3. Step 3: Decision

The status "decision" is based on the resource's attributes, on a number of external information items, and on the policies that can be applied for a resource with such attributes. Each policy evaluates a status for a resource and a result is returned to the PDP. Policies use a limited

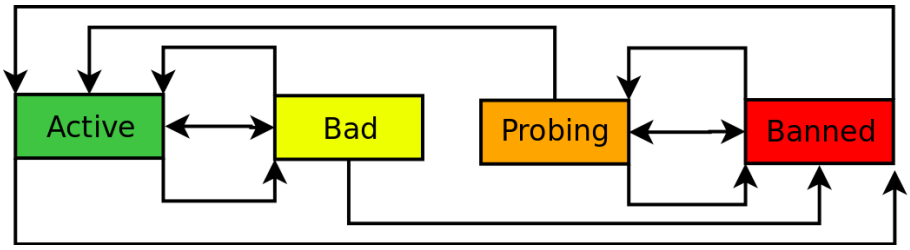

Figure 4. LHCb status and possible transitions

number of parameters, and thresholds, to evaluate a status. Together, they build up the Quality of Service (QoS) requested. A formalization of such QoS, persisted in Service Level Agreement DB, is envisaged and will complement the Resource Status System.

When multiple policies are evaluated, the PDP combines the results of the policies, and returns a single status to the PEP. The example shown in figure 5 takes a conservative approach, choosing as result the "worst" calculated.

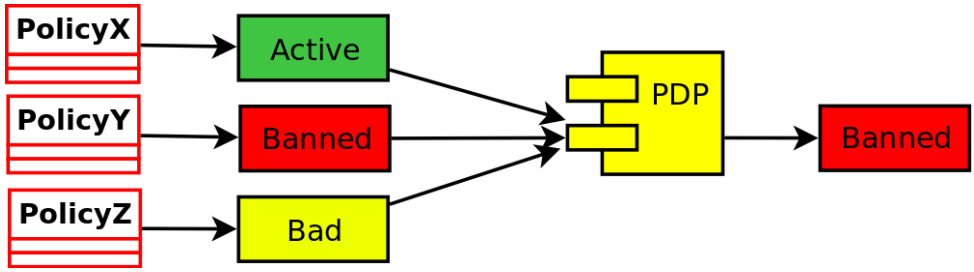

Figure 5. Simple example about combining multiple policies results

5.4. Step 4: Enforcing

In the last step, explained through the UML diagram of figure 6, the PDP returns to the PEP a decision in the form of a status. The result of each status change, together with the reason that led to that status, is persisted in the DB. The enforcing phase is specific to the Virtual Organization, and a number of plug-ins depending from the requested action can be invoked.

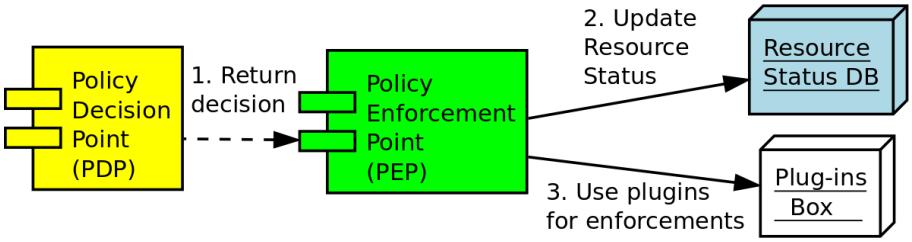

Figure 6. UML diagram of the fourth step of the cycle

Figure 7 shows the enforcing phase as has been agreed in LHCb: sites and storage elements can be banned (and unbanned) depending on the status evaluated by the RSS.

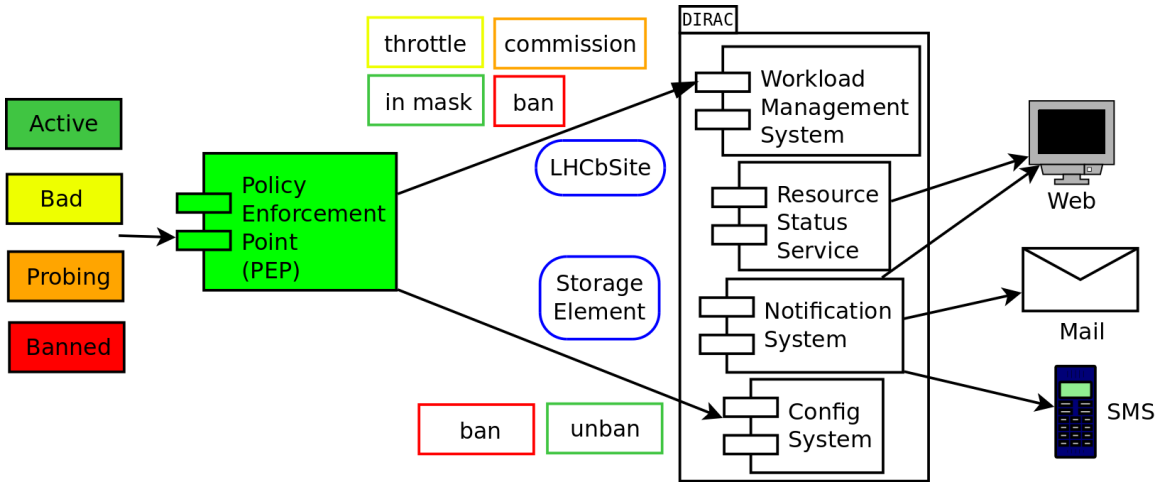

Figure 7. Example of status enforcing

6. Conclusions

In this paper we presented for the first time the Resource Status Service, used in the LHCb collaboration to monitor, compute and enforce policies that change automatically the operational status of an infrastructural grid resource. The RSS is not only a complex monitoring system: we believe it is the first tool that tries to actively use monitoring information. In that respect it is the first step towards the use of autonomic computing technologies from a VO perspective. The RSS is an active development inside LHCb and DIRAC, and the distributed computing working group expects to take advantage of its potentialities. The RSS is a new DIRAC system, ready to use by the other VOs using DIRAC.

References

[1] Andreeva J, Gaidioz B, Herrala J, Maier G, Rocha R, Saiz P and Sidorova I 2009 131–139 10.1007/978-0-387-78417-5_12 URL `http://dx.doi.org/10.1007/978-0-387-78417-5_12`
[2] Collaboration L Lhcb collaboration page URL `http://lhcb.web.cern.ch/lhcb/`
[3] Tsaregorodtsev A, Bargiotti M, Brook N, Ramo A C, Castellani G, Charpentier P, Cioffi C, Closier J, Diaz R G, Kuznetsov G, Li Y Y, Nandakumar R, Paterson S, Santinelli R, Smith A C, Miguelez M S and Jimenez S G 2008 *Journal of Physics: Conference Series* **119** 062048 URL `http://stacks.iop.org/1742-6596/119/i=6/a=062048`
[4] Abe T and Al 2010 Belle II Technical Design Report URL `http://xxx.lanl.gov/abs/1011.0352v1`
[5] Barish B and Al Global Design Effort URL `http://www.linearcollider.org/GDE`
[6] ITU-T 1995 Information technology open systems interconnection security frameworks for open systems: Access control framework Tech. rep. ITU-T
[7] Westerinen A, Schnizlein J, Strassner J, Scherling M, Quinn B, Herzog S, Huynh A, Carlson M, Perry J and Waldbusser S 2001 Terminology for Policy-Based Management RFC 3198 (Informational) URL `http://www.ietf.org/rfc/rfc3198.txt`
[8] OASIS 2005 Oasis extensible access control markup language (xacml) tc http://www.oasis-open.org/committees/xacml
[9] Burke S, Campana S, Lanciotti E, Mendez Lorenzo P, Miccio V, Nater C, Santinelli R and Sciaba A 2009 *gLite 3.2 User Guide* URL `https://edms.cern.ch/file/722398//gLite-3-UserGuide.pdf`