# Framework for GRID Metascheduling with SLAs

*Bertogna, Leandro M. - Del Castillo, Rodolfo*
*Computer Science Department, Universidad Nacional del Comahue,*
*Buenos Aires 1400, Neuquén, Argentina*
*{mlbertog, rolo}@uncoma.edu.ar*

*Abstract*
*Integration of heterogeneous resources in different administrative domains makes control and management of these environments a hard task, and this could be even worse if organizations intend to use these resources in a coordinate manner. Our goal is to simplify these labors with a policy based management schema. Policies with a high level of abstraction will be transform automatically in business rules to the right entities.*
*In this paper we define a framework and several design aspects to show how this policy based management schema can be done. Besides we will give an example of a task scheduler for a computer cluster and how the latest version of grid tools available in the market fit in this proposal.*
*Key words: Grid, SLA, metascheduling*
*Workshop de Procesamiento Distribuido y Paralelo*

## 1. Introduction

In most of organizations, computer networks computing and information resources, are critical points to do an optimal administration. With their evolution these resources become complex and singular; different type of users, different type of services, access control, software and hardware heterogeneity, require a lot of funds for maintenance. To satisfy these requirements personal with different profiles and high skills are needed. This can be seen more clearly in Grid environments, because not only they have different administrative domains but also different kind of resources, like storage, computing, visualizations, networks, etc.
Our goal is to fusion, quality of service parameters, admission control, congestion management, congestion avoidance, load balance, etc, with high level policies or business rules, in one simple, centralized management schema.
An example of these policies could be job submission from grid organizations. Policies will modify schedulers queues without knowing where the resources are located or if they are heterogeneous, and local schedulers where these resources are physically located and where the policies are enforced.
We can see that the use of rules covers a great range of administrative tasks and involves classification of users in roles, classification of services, differentiate services over time, each one of these tasks have to be tuned with packet lose, jitter, MTT, etc. These policies have to be interpreted, stored and applied in the correct locations automatically, and have to be flexible to include new requirements and future devices.
This work shows an architecture to achieve this framework based in the RFC 2753 [1]. This RFC is concerned with specifying a framework for providing policy-based control over admission control decisions.
In the rest of this paper we will analyze how can we extend most of the RFC elements to grid environments and we will propose implementations for each one. The first section will describe policy based networks elements, and grid environments, the second

section we will develop the framework and some adaptations to grid. In the next sections we will describe examples and show how different tools fit easily in this schema.

## 2. Background

At some point, it will become necessary for systems to become much easier to manage, so that they do not require significant amounts of routine administration. Ideally, they should require no routine administration tasks like allocation of file systems, shuffling of computer resources, or job scheduling. System administration as a discipline will continue to be of critical importance, though, as the function that determines the policies to be implemented by self-managing systems; these goals are under research by groups working in autonomic computing [2].

As we can see from [3] self-administering systems are critical in grid computing. Several works trying to solve this kind of problems do exist, first tries were policy based networks [4][5]. When grid turned up there were some migration works [6][7], but these ones showed the practical benefit using policies rather than how these policies are stored and managed.

### 2.1 Policy based Networks

The elements that will be the base of our framework are: the Policy Decision Point (PDP), the point where policy decisions are made and the Policy Enforcement Point (PEP), the point where the policy decisions are actually enforced.

The PEP corresponds directly to a grid node and the PDP is an entity that can be located in the policy server and can use other functionality like authentication servers, account management, information storage, etc.

Policies represent goals in the organization, in our case goals of the virtual and local organization. A translation must be made between these goals and objectives and their realization in the network. An example of this could be a Service Level Agreement (SLA), and its objectives and metrics (Service Level Objectives, or SLOs), which are used to specify services that the network will provide for a given client. The SLA will usually be written in high-level business terminology. SLOs address more specific metrics in support of the SLA. These high-level descriptions of network services and metrics must be translated into lower level, but also vendor and device independent specifications.

### 2.2 Web Services

There are experiences trying to obtain platform independence to assure a service agreement between peers.

At application level, web services is a technology widely used to achieve this independency based on a service-oriented architecture (SOA).

Grid applications are more restrictive and possess special characteristics, like complex data structures for I/O or the need to express stateful resources. To solve this aspects a set of devices like WS-Resource [8] have been proposed as a means of expressing the relationship between stateful resources and Web services, and the WS-Resource framework [9], a set of proposed Web services specifications that define a rendering of the WS-Resource approach in terms of specific message exchanges and related XML definitions. These specifications allow the programmer to declare and implement the association between a Web service and one or more stateful resources.

To accomplish some kind of contract between a service provider and a customer, such as to define the obligations of the parties, there is a specification language for service level agreements for Web Services, the Web Service Level Agreement [10]. This idea is naturally extended to grid like in the WS-Agreement [11], this draft describes an XML language for specifying an agreement between a resource/service provider and a consumer, and some works can be seen in [12].

## 3. Framework

The basic architecture proposed by the RFC 2753 says that the interaction between the components begins with the PEP. The PEP will receive a notification or a message that requires a policy decision. Given such an event, the PEP then formulates a request for a policy decision and sends it to the PDP. However, this is based on the assumption that most of the cases fall within the scenario of one administrative domain with a relative low number of resources. If we replicate this to other domains and we combine them like in grid usage, this idea will be hardly scalable. Virtual organizations [13] add another level of abstraction in the decision hierarchy, we have a local level control, the organizations that have the resource physically and a meta level, the organizations that have the resource virtually. These two levels may overlap or have conflicts.

To allow these levels of decisions to work correctly, and trying to achieve modularity and scalability, another architectural element that describes the RFC is the extension of the PDP with a local representative (LPDP). The RFC interprets this representative as a network node and this means that the PEP will first use the LPDP to reach a local decision. This partial decision and the original policy request are next sent to the PDP, which yields a final decision (possibly, overriding the LPDP). It must be noted that the PDP acts as the final authority for the decision returned to the PEP and the PEP must enforce the decision yielded by the PDP.

If we extend this behavior to grid environments and modify this concept, we will assume the LDPD as representative of the local organization that is a component of the VO. We could store local policies in the LPDP and global policies or VO policies in the PDP: both level of policies will be analyzed, but in some cases global policies will have high priority; in other cases, local policies will be independent, without interaction with the global PDP. This could reduce the policy search domain. An example of this is an authentication service: we have local users and global users, and some of these are not the same. There is no point for both decision points to have knowledge of both users, they should only know their scope instead. Once permitted access by the PDP or the LPDP the PEP just do the job, achieving task logic cohesion and improving decision-making performance.

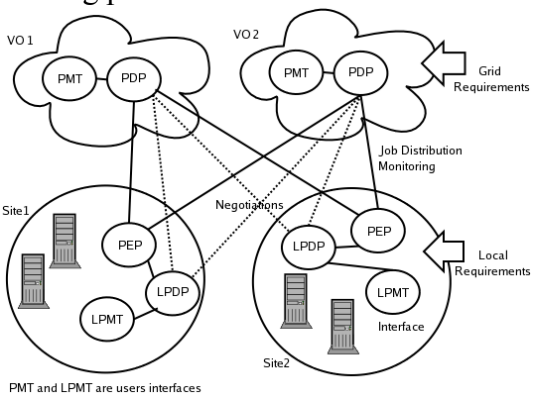


Fig.1

## 2.Mechanisms for metascheduling

We have defined the framework, but not how it will be carried out. Now we have to evaluate which mechanisms are desirable for it. Mechanisms must include support for monitoring policy state, resource usage, and provide access information. In particular, mechanisms must be included to provide usage and access information. This may be used for example for accounting and billing purposes.

Most of these mechanisms are already in use or proposed. Let us take a job submission scenario proposed on the WS-agreement draft, and we will be able to see how this procedure can be adapted to the administration framework.

## 4.1 Submitting a job

The RFC[1] says that the translation of policies into SLAs is possible. We will use WS-Agreement as a model and we will suppose that resources policies defined by administrators are already translated into

SLAs.
To use the same terminology with the WS-Agreement draft we will call these SLAs *Agreement-Template.* These are resource templates and will be store in each PDP where the resource is published.
A service provider may post an agreement template to the local PDP and grid PDPs. In this scenario, the agreement template defines the list of applications to be executed, and the software execution environment typically specified in a job submission. Service consumers are given a quality of service guarantee in terms of number of nodes and/or per node memory and storage for a specific time period. Alternatively, the guarantees can be on the completion time. A service consumer requesting a submitted job must fill in the name of the application to be executed, input and output files. In addition, a service consumer chooses the number of nodes (or any other resource requirements) that the application is guaranteed to execute on.
To submit a job, a service consumer retrieves the template from the PDP, selects the application name, and provides URL of the input and output files as well as the details of resource guarantees. The filled template is sent as an offer to the provider; in our case this provider will be a PDP. The PDP analyzes this by sharing the decision among others PDPs, VO´s PDP or LDPD where the resource is located. Finally decides whether to accept or reject the request. This may depend on the queue of jobs waiting to be processed and the current allocation of resources. The service provider answers the offer with a confirmation or a fault. In due time, the service provider processes the job.
These agreements will be stored in different PDPs for the next requirements.

## 4.2 Monitoring

One of the most important parts of this schema is task information gathering; decisions and agreement verifications are based on this data. This information is obtained from tools like the Monitoring and Discovery System (MDS)[14] component of Globus Toolkit V4. This component can streamline the tasks of monitoring and discovering services and resources in a distributed system or Grid.
The MDS framework requires that resources explicitly register with the aggregator service. The aggregator service periodically collects up-to-date state or status information from all registered Grid resources using specific information sources. In clusters, we have the GLUE [15] resource property that collects information from the schedulers and the cluster information system, for example Ganglia [16] or Hawkeye [17], and makes this information available to the user.
If we observe the task of explicit register, we have already done it by the Agreement-Template publication, as we see in the section before; this reminds us that the aggregator service is located in the PDPs.
Based on the monitoring tool information, task submitting will be done by metaschedulers. This software allows interaction with underlying resource managers in a system independent fashion. In addition, the metascheduler is a high-level abstraction for some of the concepts like “job”, a “resource reservation”, and a “scheduler”. This level of abstraction is in the VO PDP, submitting tasks to a local PEP. In grids we already have this schedulers implemented like the CSF [18] metascheduler. This metascheduler has adapters that can communicate with local schedulers OpenPBS [19], SGE [20], LSF [21], so combining their functions could enforce policies globally and locally.

## 3.Conclusion

The model discussed in this paper defines an adaptation of policy based network administration into a grid environment. This schema has several benefits such that configuration of multiple instances of the same entity is feasible, or that the configuration of different entities can be done from a single tool in a centralized manner.

Defining administration scopes, pointing out where decision points are located, and using well-defined interfaces, ensure consistent configuration across products using the same objects in their configuration.

Now we are developing a state of the art report of monitoring tools so we can test which one, or which combination of them, fits better in our schema. In the near future we plan to implement a small testbed with a meta-scheduler and a few local schedulers controls by SLAs and SLOs in both levels of control: grid and local.

## Acknowledgement

We would like to thank other members of the computer science department for their support, especially Eduardo Grosclaude.

## 4.Bibliography

[1] Yavatkar, R., Pendarakis, D. and R. Guerin, "A Framework for Policy-based Admission Control", RFC 2753, January 2000.

[2] Jeffrey O. Kephart, David M. Chess, "The Vision of Autonomic Computing", *IEEE Computer* 36(1): 41-50 2003.

[3] I. Foster et al., "The Grid2003 Production Grid: Principles and Practice", Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing, 2004.

[4] Moore, B., Ellesson, E., Strassner, J. and A. Westerinen, "Policy Core Information Model -- ", RFC 3060, February 2001.

[5] Jean-Christophe Martin,. Policy-Based Networks., Sun BluePrints. OnLine, October 1999.

[6] Czajkowski, K., Foster, I., Kesselman, C.,Sander, V. and Tuecke, S., SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. in 8th Workshop on Job Scheduling Strategies for Parallel Processing, (2002).

[7] Dumitrescu, C. and I. Foster. Usage Policy-based CPU Sharing in Virtual Organizations. in 5th International Workshop in Grid Computing. 2004.

[8] Modeling Stateful Resources With Web Services. http://www-106.ibm.com /developerworks/library/ws-resource/ws-modelingresources.pdf

[9] The Web Services Resource Framework. http://www-106.ibm.com/developerworks /library/ws-resource/ws-wsrfpaper.html

[10] IBM, WSLA Language Specification, Version 1.0. 2003.

[11] K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke and M. Xu.Agreement-based Grid Service Management (OGSIAgreement) Version 0. https://forge.gridforum.org /projects/graap-wg/document/Draft_OGSI-agreement_Specification/en/1/Draft_ OGSI-Agreement_Specification.doc

[12] Providing Data Transfer with QoS as Agreement-Based Service, H. Zhang, K. Keahey,W. Allcock. 2004.

[13] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15(3), 2001

[14] MDS, http://www.globus.org

[15] GLUE, http://www.cnaf.infn.it /sergio/datatag/glue/

[16] Massie, M., B. Chun, and D. Culler. The Ganglia Distributed Monitoring: Design, Implementation, and Experience. in ParallelComputing. May 2004.

[17] Hawkeye monitoring tool, http://www.cs.wisc.edu/condor/hawkeye/

[18] Community Scheduler Framework. http://sourceforge.net/projects/gcsf/

[19] OpenPBS Project, A Batching Queuing System, Software Project, Altair Grid Technologies, LLC, www.openpbs.org.

[20]Sun Grid Engine. http://gridengine.sunsource.net/

[21] LSF Administrator's Guide, Version 4.1, Platform Computing Corporation, February 2001.