

Uniform Job Monitoring using the HPC-Europa Single Point of Access

F. Guim¹, I. Rodero¹, J. Corbalan¹, J. Labarta¹
 A. Oleksiak², T. Kuczynski², D. Szejnfeld², J. Nabrzyski²
 Barcelona Supercomputing Center¹,
 {francesc.guim, irodero, julita.corbalan, jesus.labarta} @ bsc.es
 Poznań Supercomputing and Networking Center², Noskowskiego 10,
 60-688 Poznań, Poland
 {ariel,docentt,dejw}@man.poznan.pl

Abstract—Job monitoring in Grid systems presents an important challenge due to Grid environments are volatile, heterogeneous, not reliable and are managed by different middlewares and monitoring tools.

We present the infrastructure that we have designed and implemented in the HPC-Europa European project that allows uniform access to job monitoring information from different virtual organizations. The presented system abstracts user to the complexities of the underlying systems of each middleware. The API that each center has to implement for providing access to its job monitoring information is explained. Finally, we show all the features that user can use in the portal to personalize his/her monitoring environment: how and which information has to be presented.

Index Terms—Grid computing, job monitoring, Grid portals, HPC-Europa

I. INTRODUCTION

Grids environments allow users submitting jobs to many different centres and resources across different administrative domains. Each of those centres may belong to different Virtual Organizations (VO) [1] and may have different resources, different middleware and different characteristics. Job monitoring provide functionalities that allow users taking actions and monitoring jobs submitted to them.

Usually, each HPC center has installed its own monitoring tool or software components that allow tracking jobs. Section II presents some of the most relevant monitoring tools that can be used for monitoring. They provide different mechanisms for query job information (LDAP, Open Grid Services, TPC-IP, etc). Moreover, they have different properties, and the way how client accesses to them are completely different. Also brokers provide functionalities for accessing information about the jobs that it manages and controlling them.

This paper has been supported by the “HPC-Europa” project and by the Spanish Ministry of Science and Education under contract TIN2004-07739-C02-01.

When user has to access to jobs information of a given center, he/she must know how to query the job monitoring tool or broker that is running on it, which information is provided and which is the format. The complexity of working with such a complex and heterogeneous systems, where several job monitoring entities are running, will make user desist of using them. We can not expect end user to use many different tools for achieve the same final object: tracking their submitted jobs.

We strongly support that Grid features must be accessible in a uniform and easy way. Abstracting the complexity of the underlying monitoring systems they will come more usable and transparent. This point of view is also applicable to the job monitoring.

In this paper we present how we have achieved such goal in the HPC-Europa project with the single point of access (SPA). Using the provided Grid portal user is able to monitor all the jobs submitted to the different middleware in a uniform way. Before designing the interface presented in this paper, in order to reach this final goal, we carried out a deeper study of the state of the art of monitoring. We studied a set of different monitoring tools that we considered more relevant and their interfaces (more detailed information is provided in the next section). We studied the interfaces that brokers such GridWay[2], schedulers such LoadLeveler[3][4] and middleware such Globus [5][6] provide for job control and monitoring. The recommendations for the GGF DRMAA Group (Distributed Resource Management Application API) concerning this topic [7] were also studied.

We present the architecture of the SPA and the description of the components concerning job monitoring. Moreover we present how the information is provided by one of the centers that are accessible in the portal. We show how the eNANOS infrastructure allows monitoring of jobs submitted to the Barcelona Supercomputing Center (BSC-CNS [19]).

The rest of the paper is organized as follows: Section II presents the monitoring tools that we consider more representative; section III presents the architecture of the Single Point of Access, its main features and characteristics; Section IV presents the global architecture for job monitoring of the SPA and the generic interface that we have designed;

Section V shows the monitoring features that are available to the user in the HPC-Europa portal (which uses the presented interface); Section VI present how the eNANOS middleware have implemented such interface; and finally, section VII present the conclusions.

II. RELATED WORK

There are several tools or services that allow monitoring jobs in Grids. The MDS is the Grid Information Service provided in the Globus Toolkit [8]. It supports monitoring sensors to register to the Information Service. Two versions of MDS are currently available; MDS-2 uses the Open LDAP protocol whereas MDS-3 uses the Open Grid Services Infrastructure. This architecture provides a hierarchy over different administrative domains to form Virtual Organizations (VO). MDS has no fine-grain information access policies.

Under the GridLab [9] European research project the monitoring tool Mercury has been developed. Like MDS, Mercury is a hierarchal monitoring service, organized with Local Monitors at the host end, and Main Monitors. Mercury allows several authentication methods: by IP, through GSS-API (Generic Security Service, [rfc1508]), or by new modules that can be added. Once authenticated, the user has access to the metrics following a resource-level authorization. Many metrics are provided, allowing an accurate resource-oriented monitoring. Mercury is the only tool to provide such fine-grain access policies.

Another representative monitoring tool is the Network Weather Service (NWS) [10]. It is a monitoring system designed for short-term performance forecasts. NWS provides a set of system sensors monitoring end-to-end TCP/IP performance, available CPU percentage, and available memory. Based on collected data, NWS dynamically characterizes and forecasts the performance of network and computational resources. It is no possible to have a hierarchical NWS system, and so it cannot work through different domains alone. However, it can be integrated in other tools, since it uses LDAP, like MDS.

The Ganglia [11] project provides a distributed monitoring system for high-performance computing systems such as clusters and grids. It relies on a multicast-based listen/announce protocol to monitor state within clusters and uses a hierarchy amongst representative cluster nodes to federate clusters and aggregate their state. Ganglia also comes with a web front-end to visualize the metrics evolution.

As a representative job monitoring tool, we can find OCM-G (OMIS-Compliant Monitoring system for the Grid) [12][13]. It is an application monitoring tool, which is part of the Crossgrid project [14]. Crossgrid is also providing G-PM (Grid-oriented Performance Measurement tool) which is a graphical performance analysis tool that allows requesting standard performance metrics as well as user-defined metrics at runtime. G-PM allows a distributed on-line analysis of raw data to provide user-defined metrics at runtime, whereas usual approaches (Pablo [15], Paraver [16], etc) only support a

centralized and offline analysis.

III. THE SINGLE POINT OF ACCESS

One of the HPC-Europa project objectives is to design and develop the portal, and integrate it with the middleware required for the realization of the Single Point of Access. To this end, the HPC-Europa Grid Portal, based on the GridSphere portal framework [20] including its GridPortlets mechanism, is being developed. The portal will provide transparent, uniform, flexible and intuitive user access to HPC-Europa resources. This portal will hide the underlying complexity and heterogeneity of these resources and the access to them.

The main requirement of this activity is to build a portal that provides a uniform and intuitive user interface. However, most of the HPC centers have already deployed their own site-specific HPC and Grid infrastructure. Therefore, an additional requirement is to keep the autonomy of HPC centers allowing them to use their favorite middleware, local policies etc. For instance, there are currently five different systems that provide a job submission and basic monitoring functionality in the HPC-Europa infrastructure: eNANOS resource broker [18], GRIA middleware [21], GRMS – Grid Resource Management System [22], JOSH – Job Scheduling Hierarchically [23], and UNICORE [24]. Additionally, eNANOS, GRMS and JOSH use the Globus Toolkit [5] to access underlying resources provided for the HPC-Europa infrastructure.

The Single Point of Access will provide two kinds of interfaces to application users. First, a generic interface has been developed, which can be used by all users for most of their batch applications. To this end, uniform interfaces are provided for the most relevant Grid functionality identified from a requirements analysis of the centers. The following key functionality has been foreseen to be required for the realization of the SPA: job submission, job monitoring, resource information, accounting, authorization, and data management.

In addition to the generic interface, application-specific portlets are being developed to allow users to manage more complex (e.g. interactive or requiring many specific input parameters) applications in a straightforward manner.

In order to provide end-users with transparent access to resources, we developed a mechanism responsible for the management of uniform interfaces to diverse Grid middleware. Using this mechanism the Single Point of Access enables dynamic loading of components that provide access to the functionality of specific Grid middleware through a single uniform interface. These components are called plug-ins in this context. These uniform interfaces are based on standards where possible (e.g. JSDL [25] for job submission – see [28] for more details) and functionality provided by Grid middleware deployed in HPC centers.

From the end-user perspective, a uniform Graphical User Interface is provided that is common for all systems deployed in the HPC-Europa infrastructure. This GUI can be dynamically adapted to particular systems and still keep the same look and feel. Only slight modifications such as disabling fields, limiting lists of values etc. are allowed.

To this end, we have implemented the ability to check the functionality of every single system by retrieving the capabilities of site-specific plug-ins. These descriptions of the implemented capabilities are returned in the form of the appropriately constrained general XML schema. A plug-in returns two descriptions: a description of the methods it supports and a description of data structures (e.g. job description). In [28] this mechanism is described in more detail taking the Job Submission Portlet as an example. The general architecture of the portal part of the SPA is illustrated in Figure 1. All portlets included in the SPA are JSR168-compliant [26] with separated business logic (portlet services) based on the Spring framework [27]. They are therefore portable, meaning that they can be deployed in portal containers other than GridSphere.

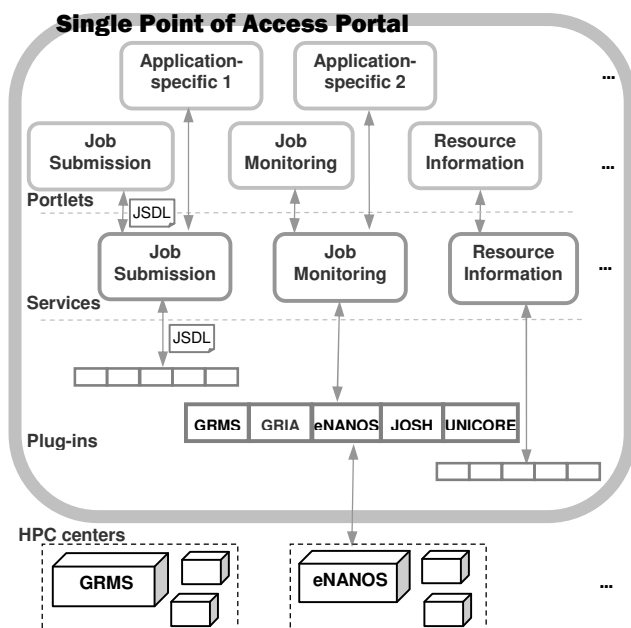


Figure 1: Architecture of the Single Point of Access portal

Next sections present the main features concerning the job monitoring that are provided in the SPA portal to the end user. We also present a detailed description of the main components that are part the job monitoring infrastructure of the portal.

IV. JOB MONITORING ARCHITECTURE

This section provides an overview about the internals of the job monitoring architecture that has been developed in the scope of the project. We present the different components that take part to the monitoring process and their main functionalities. The common job monitoring interface that is implemented by all the centers is also presented.

Figure 2 presents a global overview of the architectural components. The monitoring has three main parts: the job monitoring portlet, the job monitoring service and the generic monitoring interface that all plug-ins implement.

A. The job monitoring portlet

The monitoring portlet is the responsible of showing the

monitoring information and process all the actions required by users (See section V).

As a portlet it has two modes: the edit view and the main view. The **edit view** allows the user to set all the parameters and visualization characteristics (See section V). All the users setting are saved in a persistent XML document that will be reloaded each time that user uses the monitoring system by first time in a session. There are a set of predefined profiles that can be loaded from a set of XML documents that contains a predefined configuration values. The portlet has an XML Schema that defines which fields that contain monitoring information that will be available to the user. This XML Schema defines all the information that currently can be provided by the monitoring system. The configuration mode does not depend on plug-ins neither services.

The **main view** shows the information of the jobs that user has submitted through the HPC-Europa SPA. This is presented depending on the user settings (set in the edit view) and depending on the information that each center provides.

The list of jobs is retrieved from the job monitoring service that will collect them from the plug-ins. The job information is gathered also from each plug-in. Plug-ins return an XML based on a common XML schema that contains the fields that will be shown on the monitoring page. As not all the plug-ins may be able to fulfill all the information that can be provide the XML, they are able to return a restriction of such XML Schema without all those fields that they can not provide. These restricted XML Schemas will be used in the monitoring page to tell to the user when a given field can not be provided by the center. As an example, Figure 3 shows the list of the jobs that user has submitted. The center HLRS and Poznan do not provide information concerning job progress, and they have specified it in the restricted XML Schema.

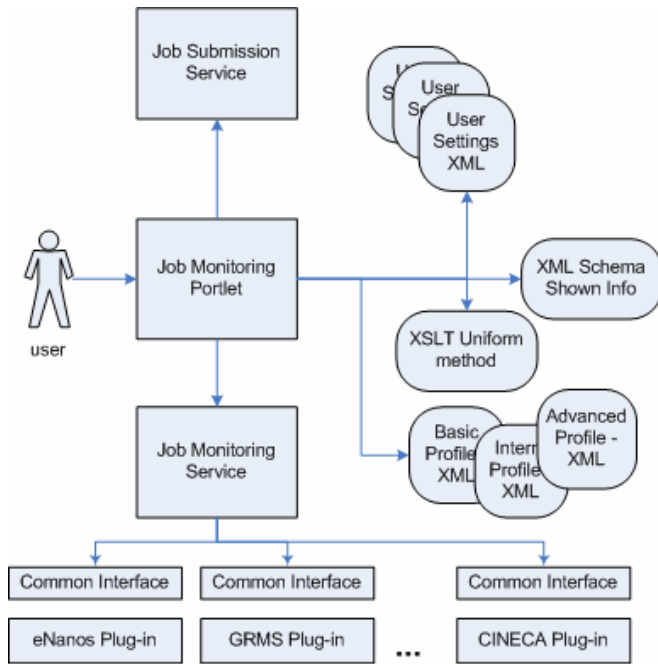


Figure 2: Global architecture for monitoring

Job id	State	Center	StartTime	Progress	FinishTime	Action:
6@113	GRID_QUEUED	BSC	2005-11-12 T19:20:8	10 MIPS	-	<input type="checkbox"/>
7@113	RUNNING	Poznan	2005-11-12 T19:21:15	No provided.	-	<input type="checkbox"/>
4@113	RUNNING	Poznan	2005-11-12 T19:25:13	No provided.	-	<input type="checkbox"/>
5@114	RUNNING	HLRS.	2005-11-12 T19:27:56	No provided.	-	<input type="checkbox"/>
2@116	FINISHED	BSC	2005-11-12 T19:20:12	-	2005-11-12 T19:22:4	-
1@118	FINISHED	Poznan	2005-11-12 T19:19:12	-	2005-11-12 T19:20:2	-
3@117	FINISHED	HLRS	2005-11-12 T19:15:10	-	2005-11-12 T19:16:12	-

Figure 3: Table with summarized information

B. Job Monitoring Service

The job monitoring services works in the same way as the other services of the portal. It receives information from the job monitoring portlet and proxy the query to the appropriate center plug-in. For example, it can call the method *getJobPS* (explained in next subsection) to the plug-in of the center Barcelona Supercomputing Center for gather information of a given job submitted to the center.

C. Plug-in monitoring methods

As explained in section III, the plug-ins centers implement a common interface that provides same functionalities. This section provides a description of all the methods related job monitoring that has been added to the plug-ins interfaces and that all of them must implement.

1) XML *getSupportedMonitoring()*

This function returns the XML schema document that contains the monitoring information that the center supports. This schema as have been explained is a restriction of a common XML Schema. In case that the plug-in can not provide a given field or information, such as Job type or current state, the XML Schema won't have the definition of the element that conveys such information. For instance if the

information of the current time when the job is PREPROCESSING can not be provided by the center GRIA, it will return the XML Schema without this element inside the element CurrentState. Figures 4, 5,6 and 7 show some parts of this schema.

2) XML *setJobFilters(XML Filter, SPAUserData credentials)*

This function allows retrieving a set of given jobs ids for the given user that matches the filters specified in the XML. This function allows avoiding complexity of the portlet, avoiding no necessary communication and provides a powerful way for query the job information. We have currently defined four different filters:

- The first one is compulsory and must be implemented by accepted by all the centers. This returns all the jobs of the user.
- The second one allows filtering by state. This filter is optional.
- The third allows filtering all those jobs that are not active. Active jobs, are those jobs are not FAILED, FINISHED or CANCELED
- The last one allows filtering by submission date. If user has been using the portal for some month it makes sense to allow him to visualize on those jobs that have been submitted the last week.

Figure 4 shows the structure of the XML documents that contain the filters.

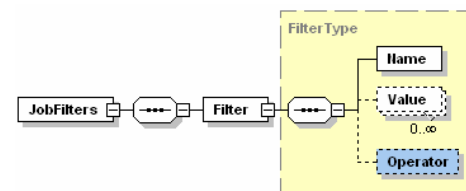


Figure 4: XML Schema for setJobFilters

3) String *GetJobInfo(String jobId, SPAUserData credentials)*

Given the user identification, his/her credential and a job id, the method returns the basic information for the given job: Job status, job start time in the local system, the name of the submitted file name and so on. This information follows the XML Schema show in the following figure.

This information is used for generate basically the summarized view explained in section V.

4) public XML *GetJobPS (String jobsList, String SiteID, SPAUserData credentials)*

The plug-in returns all the jobs information based on the provided jobs list for the given user. In this case the information for each job is more detailed and specific. This is the basic information returned by the *getJobInfo* plus more status specific information. For example if the job is running the plug-in may return the status of each process its process. Figures 5 and 6 shows a part of the XML Schema that returned XML follows.

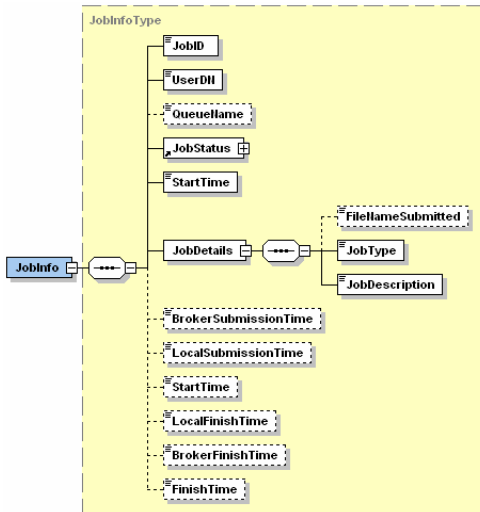


Figure 5: XML Schema for get JobInfo

Something important is that not all the centers must be able to return all the information specified in the XML Schema. It has been defined for uniform the format and data types for the data that plug-ins can return. Centers can specify which information they are returning using implementing the explained function *getSupportingMonitoring*.

5) *String GetJobExitCode(String jobID, SPAUserData credentials)* and *String GetJobStatus(String jobID, SPAUserData credentials)*

These are the last functions that allow retrieving information from the center. They return the exit code and the status for a given job respectively.

6) *Vector getSupportedActions()*

As not all the centers are available to provide all the actions that can be done over a job (such as cancel or resume). They can specify which of them are supported with this method. The monitoring portlet will allow taking a given action in a given job if the center where such job is running allows it.

7) *Methods for taking actions over jobs*

The following methods allow modifying the state of the job:

- *String CancelJob(String jobID, SPAUserData credentials)*
- *String ResumeJob(String jobID, SPAUserData credentials)*
- *String SuspendJob(String jobID, SPAUserData credentials)*

As an example of a possible sequence of interaction between the portlet and the service (that would call to the plug-in) could be: user load the monitoring page of the portlet; the portlet has to list the user jobs that he/she has submitted through the portal to the centers; it retrieves such list calling the set job filters for each center with the appropriate filters (based on the user configuration). Once the portlet has the list of jobs, it can retrieve the information of each job calling the *getJobPS* or *getJobInfo* for each job in the correspondent center. The portlet is now available to show the information of each job to the user.

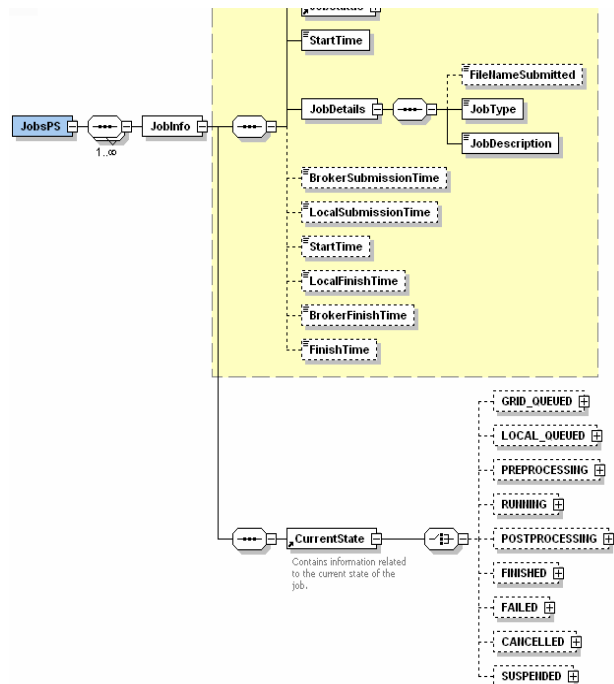


Figure 6: XML Schema for the GetJobPS

Section VI presents the implementation of the monitoring methods in the plug-in of the Barcelona Supercomputing Center. We show how the information is provided by the eNANOS broker to SPA

V. JOB MONITORING USER AVAILABLE FEATURES

A. Monitoring information

Using the monitoring page user is able to control all the information concerning the jobs that has submitted to all the centers. Which information is shown and how this information is shown depends on the settings that she/he has set on the configuration mode explained on the next section. Something important is that the information shown in each job depends on the center that it has been submitted, because not all the centers support all the enumerated fields on the next subsection paragraphs. For example, as has been explained, a given center may not support the field estimated remaining when the job remains grid queued but others not. When a job has some field with an empty value it is shown as not available.

Using this interface user is also capable of take actions over the jobs and gathers more detailed information of a given job. User can:

- Select all or a subset active jobs (those jobs that are running, grid queued and local queued), and cancel, suspend or resume them. In case that the job belongs to a center that does not support a given action the action will be ignored.
- Require more detailed information of a given job. This feature allows getting more specific information of a

given job, for instance a list of processes and threads that it is running and their status and progress.

- Order jobs based on different criteria: order by submission date, center, state, job id and finish time.

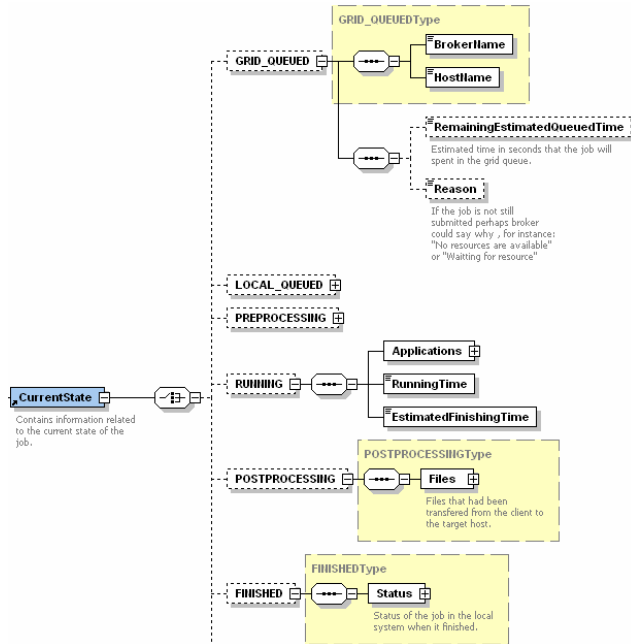


Figure 7: Part of Current State element of the GetJobPS

B. View personalization

User is able to set its own environment and decide what he/she wants to see and how it has to be presented. For satisfy such user requirements, we have implemented a powerful configuration functionality that allows to him/her to set such characteristics. The structure of configuration is divided in five parts:

1) Loading predefined configurations

We have defined predefined configurations that allow user to load different configurations depending on monitoring requirements that he / she has on given moment. Each configuration has associated a set of predefined setting values that specify the information to be shown and how. Three configurations are currently available:

- The *basic* configuration is intended to be used for those users that do not need to access to a fine grain detail jobs information. This configuration tries to show job information in such a way that tries to abstract the Grid architecture and concepts that underlies the portal. It is intended to behaves as the Unix command “ps” for the jobs that user has submitted through the portal.
- The *intermediate* configuration is intended to be used for those users that do not require access to fine grain detail jobs information, but that are aware of the characteristics of the Grid and have knowledge of the Grid concepts.
- The *advanced* configuration is intended to be used for those users that are experts. We presuppose that they expect to see fine grain information about the jobs that

they have submitted.

Each configuration has not only associated which information has to be shown, it also has associated the view.

2) Setting global filters

Filters allows user to filter which jobs have to be shown in the main monitoring page based on several criteria. There are many situations where such functionality may be useful. For instance, user may only see jobs that has submitted the last week or month, or may only want to see jobs that are running or finished. The current defined filters are:

- User can filter based on the state of the job, for instance he/she can specify that jobs that are running, finished, failed and completed have to be shown, but those that are local queued not.
- User can filter based on the submission date, for instance filtering those jobs that have been submitted before a given date.
- User can filter based on the finish date.
- User can filter those jobs that are not active. Active jobs are those jobs that are running, finished, completed and failed.

3) Specifying fields to shown

The information associated to jobs is represented by fields. Each job has a set of fields that convey information concerning it. We have defined two kinds of fields: those that are common to all the jobs, and those that are related to the states. Depending on which is the state the job it may make sense to provide some information or not. For example, if the job is running we may provide the estimated remaining running time, but if the job is finished. Table I show the common fields available for each job and table II shows the available job fields associated to the states. User is able to select which fields have to be shown in the monitoring page when the jobs are listed.

TABLE I
COMMON JOB FIELDS

Filename submitted
Job type (depends on the center, f.e.: parallel or sequential)
Job description (introduced by user in the submission portlet)
Current Start Time (time that job has started running)

TABLE II
JOB FIELDS ASSOCIATED TO THE JOB STATE

State name	Fields
Grid Queued	Broker name, hostname, Remaining estimated queued time, Reason why it still remains queued.
Local Queued	Local Resource Manager Name (f.i: OpenPBS), Local Id (id that the local system has assigned to the job), Host,

	Remaining estimated queued time, Reason why it still remains queued.
Pre-processing	Number of fields to be transferred and number of transferred files.
Running	Number of applications, running time, Estimated Finishing time, Host name, job progress (information that shows how is progress of the application)
Post-processing	Same fields as pre-processing.
Finished	The status of the job when it has finished.
Failed, Cancelled, and Suspended	Reason why the job it is in such state.

4) Selecting the mode view

The previous explained features allow to the user to select which kind of information is shown. User can choose a set of views that allows visualizing job information grouped and organized based on different criteria:

- Summarized job list. It will show a single table with all the jobs that user has submitted to the Grid. Fields like job status, submission date, finish date (empty if it is already running or queued) and center. User can order job lists depending on each of those fields.
- Information grouped by jobs states. It will show a set of tables grouping the jobs by their state. In this view more state specific information is shown. Each job list provides the common fields plus the fields that are associated to the state of the job.

The basic configuration has associated the first view and the intermediate and advanced configurations have both views associated.

5) Auto Refresh frequency

User is able to set which is the frequency that the shown information has to be automatically updated.

VI. THE eNANOS MONITORING ARCHITECTURE AND THE eNANOS PLUG-IN

In the eNANOS team [17] we are working in coordinate the different layers involved in the execution of a Grid job, from the Grid (through the SPA Portal) to the CPU scheduling layer. To perform it efficiently we need to provide mechanisms to obtain precise information with fine-grain monitoring tools.

The Grid portal of the HPC-Europa SPA obtains the monitoring information from the BSC center through the GetJobPS method that implements the plug-in and the eNANOS Resource Broker [18]. Since the operation interface required by the portal is not exactly equivalent to the current eNANOS interface, we have implemented a gateway. The eNANOS Broker gateway is a GT3 Grid Service, accessible through its own client and has two basic functions:

- Localize the broker service
- Translate the portal operations to the eNANOS interface (including JSDL, requirements, etc.)

The components and the communications between them are shown in Figure 8.

The eNANOS Plug-in implements the required methods by specified in section IV: the `getStatus`, `getJobInfo` and `getJobPS` methods and so on. To perform these actions the plug-in contacts to the Broker Gateway and provides the credentials of the user to it. The plug-in methods localize the Broker and invoke its own specific methods. The plug-in is also responsible of translating the job statuses returned by the broker to the SPA specific status (e.g. the Broker state STAGE-IN is translated to the PREPROCESSING state of the SPA).

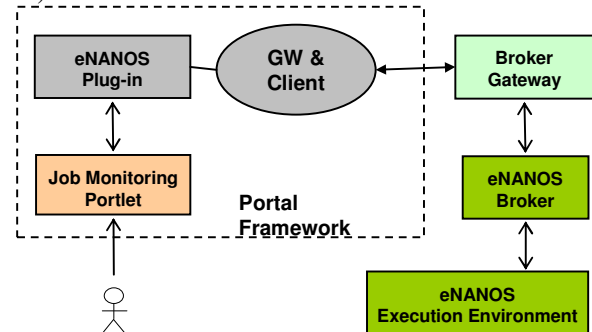


Figure 8: Portal-middleware communication schema

The most important change performed to the eNANOS Broker to give support to the monitoring infrastructure of the HPC-Europa SPA is the implementation of the `getJobInfo` and `getJobPS` methods. Therefore, the eNANOS Resource Broker returns to the SPA infrastructure a XML document with the monitoring information of the underlying level. This XML document follows the schema of the JobInfo described in previous sections. It includes several data related to the local execution systems. Actually, when a job is in the RUNNING state, the CurrentState element includes a set of applications with their respective processes and threads. This information is very important to know the behavior of the job since our project is targeted to parallel HPC applications, MPI and MPI+OpenMP as well.

The eNANOS Execution Environment is able to provide the fine-grain information about the execution of the jobs that is specified in the JobInfo schema. In the Broker it is added some information related to the Grid level, for instance the job GridID or the whole history of the job in the broker system. It also provides information about the progress of the applications. Currently there is available a prototype which allows the user to instrument statically the applications in order to obtain dynamic information about the progress of the applications. This functionality is under development since we want to provide a mechanism to show the progress of the applications dynamically without modifying the code.

We are working on providing dynamic information about the performance of the applications. We have planned to implement an API which allows the user to specify the metrics in which are interested. This mechanism should be as generic as possible and is being implemented with a similar interface as the progress indicators. We expect to provide such information in the HPC-Europa monitoring portlet in a close

future. The unique required change for provide it will be to add the necessary extensions to the current XML Schema.

VII. CONCLUSION

We have presented how using the SPA HPC-Europa job monitoring infrastructure user is able to monitoring jobs belonging to different systems. We have also shown that it provides a uniform access to job information while the underlying systems use different middleware, resources and monitoring mechanisms. It has been demonstrated that heterogeneous information coming from different administrative domains or different virtual organizations can be unified, abstracting user of technical and complex characteristics.

The global architecture of the SPA has been presented, with a specific emphasis of its components that concerns to job monitoring. Also the common job application monitoring interface that each of the underlying systems has to implement in their plug-ins is described. The extensibility of the system and the generality of the API are important characteristics. Using the XML documents for convey the monitoring information has allowed us to describe the available information adapting to the information that systems can provide. The described XML Schema will allow extending the information provided in case that new functionalities or features are required. The proposed schema can provide from very general job monitoring information, such as the job state, till very specific, such as the state of all the threads for a given process of a given job.

If an organization wants to provide job monitoring of its system to user, it just should have to implement the common methods specified in the API in its plug-in. As an example, it has been explained how such interface is implemented in the Barcelona Supercomputing Center through the eNANOS system.

The shown functionalities that the architecture provides to the user allow not only the access to a simple set of job monitoring information. They allow user to personalize the view, deciding which information has to be shown and how. User can specify filters that will be applied to the information that the system can provide. This is a powerful feature that will allow user to access only to the important that she/he requires.

REFERENCES

- [1] Virtual organization, A Mowshowitz - Communications of the ACM, 1997
- [2] A Framework for Adaptive Execution on Grids , E. Huedo, R.S. Montero and I.M. Llorente, Journal of Software - Practice and Experience. Vol. 34, pp. 631-651, 2004
- [3] Bernd Kallies and Wilhelm Vortisch. LoadLeveler, the batch queuing system of IBM. In Wolfgang Baumann, Bernd Kallies, Hinnerk Stüben, and Wilhelm Vortisch, editors, The HLRNQuickstart Guide, chapter 7. HLRN, Berlin, 2003.
- [4] Using and Administering LoadLeveler - IBM LoadLeveler for AIX 5L - IBM - Can be found at www.ibm.com
- [5] Globus: A metacomputing infrastructure toolkit, I Foster, C Kesselman, J Intl - International Journal of Supercomputer Applications, 1997
- [6] UNICORE: A Grid Computing Environment, DW Erwin, DF Snelling - Concurrency and Computation: Practice and Experience, 2002
- [7] GGF DRMAA Group. More information can be found at: <https://forge.gridforum.org/projects/drmaa-wg>
- [8] MDS is the Grid Information Service provided in the Globus Toolkit [fick97]. It supports monitoring sensors to register to the Information Service. Two versions of MDS are currently available; MDS-2 uses the Open LDAP protocol whereas MDS-3 uses the Open Grid Services Infrastructure.
- [9] GridLab site can be found at www.gridlab.org
- [10] The network weather service: a distributed resource performance forecasting service for metacomputing . References Rich Wolski, Neil T. Spring and Jim Hayes., Journal: Future Generation Computer Systems, Volume: 15, 1999
- [11] The ganglia distributed monitoring system: design, implementation, and experience. Matthew L. Massie, Brent N. Chun and David E. Culler., Parallel Computing 2004
- [12] Performance Evaluation and Monitoring of Interactive Grid Applications , Bartosz Balis, Marian Bubak, Włodzimierz Funika, Roland Wismuller, Marcin Radecki, Tomasz Szepieniec, Tomasz Arodz and Marcin Kurdziel, November 2004
- [13] The CrossGrid Performance Analysis Tool for Interactive Grid Applications , Marian Bubak, Włodzimierz Funika and Roland Wismüller., year 2002
- [14] Crossgrid site can be found at <http://www.crossgrid.org>
- [15] "Scalable Performance Analysis: The Pablo Performance Analysis Environment" DA Reed, RA Aydt, RJ Noe, PC Roth, KA Shields, BW ... - Proceedings of the Scalable Parallel Libraries Conference, 1993
- [16] "PARAVER: A Tool to Visualize and Analyse Parallel Code" V Pillet, J Labarta, T Cortes, S Girona Proceedings of WoTUG-18: Transputer and occam Developments
- [17] I. Rodero, F. Guim, J. Corbalán, J. Labarta. eNANOS: Coordinated Scheduling in Grid Environments. Parallel Computing (ParCo) 2005. Málaga, Spain. 13-16 September 2005.
- [18] I. Rodero, J. Corbalán, R.M. Badia, J. Labarta. eNANOS Grid Resource Broker. P.M.A. Sloot et al. (Eds.): EGC 2005, LNCS 3470, pp. 111-121. Amsterdam, 14-16 February 2005.
- [19] More information concerning the Barcelona Supercomputing Center can be found at: www.bsc.es
- [20] The GridSphere portal framework, <http://www.gridsphere.org>
- [21] GRIA project, <http://www.gria.org/>
- [22] Grid Resource Management System (GRMS), <http://www.gridlab.org/grms>
- [23] JOSH, <http://gridengine.sunsource.net/josh.html>
- [24] UNICORE, <http://www.unicore.org>
- [25] JSDL, <https://forge.gridforum.org/projects/jsdl-wg/>
- [26] JSR 168, <http://jcp.org/en/jsr/detail?id=168>
- [27] Spring, <http://www.springframework.org/>
- [28] A. Oleksiak, A. Tullo, P. Graham, T. Kuczyński, J. Nabrzyski, D. Szejnfeld, T. Sloan, "HPC-Europa: Towards Uniform Access to European HPC Infrastructures". *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing Grid2005, Seattle, USA, 2005.*