EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
European Laboratory for Particle Physics

Internal Note/OFF
ALICE reference number
ALICE-INT-2001-27 V 1.0
Institute reference number

Date of last change
2001-08-20

EnginFrame: the computing portal to "the GRID" to be used in the event production for the ALICE Physics Performance Report

Authors:

G. Andronico[1], A. Badalà[1], R. Barbera[1,2], P. Belluomo[1], P. Buncic[3], E. Cangiano[1], F. Carminati[2], P. Cerello[7], D. Di Bari[4,5], A. Falzone[6], A. Forte[7], L. Gaido[7], A. Guarise[7], E. Lopez Torres[7], G. Lo Re[1, 2], S. Lusso[7], M. L. Luvisetto[8], A. Masoni[9], D. Mura[9], A. Palmeri[1], G. S. Pappalardo[1], B. Platania[1], A. Pulvirenti[1, 2], F. Riggi[1, 2], C. Rocca[1], A. Rodolico[6], G. Sava1, Y. Schutz[10], M. Sitta[7], R. Turrisi[11], G. Ugolotti[6]
on behalf of the ALICE GRID Project.

1INFN, Sezione di Catania - Italy
2Dipartimento di Fisica e Astronomia dell'Università di Catania - Italy
3CERN - Geneva - Switzerland
4INFN, Sezione di Bari - Italy
5Dipartimento di Fisica dell'Università di Bari - Italy
6NICE s.r.l. - Camerano Casasco (AT) - Italy
7INFN, Sezione di Torino - Italy
8INFN, Sezione di Bologna - Italy
9INFN, Sezione di Cagliari - Italy
10SUBATECH, Nantes - France
11INFN, Sezione di Padova - Italy

Abstract:

The computing portal EnginFrame, embedding the currently available "GRID" services, is presented. Its adoption in ALICE in view of the huge event production foreseen for the realization of the Physics Performances Report is also proposed and discussed with real examples.

1 Introduction

The ALICE [1] Off-line Project [2] has a strong commitment to use the results of the just started DataGRID Project [3] which aims at setting up, in Europe, a geographically distributed computing fabric following the principles of the so-called "GRID" [4]. One of the ALICE use cases within the DataGRID Project is represented by the realization of the MonteCarlo simulations needed for the completion of the ALICE Physics Performance Report being prepared this year. In order to carry out the production of these simulated events on a distributed environment, the ALICE Off-line community has recently taken the decision to exploit some of the currently available "GRID" services included in the `Globus` Toolkit [5] such as the GRID Security Infrastructure (GSI), the `Globus` Resource Allocation Manager (GRAM), the GRID Index Information Service (GIIS), and the GRID Resource Information Service (GRIS).

In this framework, and profiting from the invaluable experience acquired on `Globus` in the last few months within the GRID Project [6] of the italian Istituto Nazionale di Fisica Nucleare (INFN), we have interfaced `EnginFrame` [7], a last generation web computing portal, with all the above "GRID" services in their present status. In doing that, we have implemented a collection of "services" defined on top of the corresponding `Globus` commands/services [9].

In this note we describe `EnginFrame`, its services and their functionalities, and its application to the above ALICE use case. The paper is organized as follows. Section 2 is dedicated to a technical overview of the computing portal and will describe its main principles. General "GRID" services are described in Section 3, while ALICE specific "services" are described in Section 4. Summary and conclusions are drawn in Section 5.

2 Technical overview

`EnginFrame` was designed to address and solve some typical problems of the Technical and Scientific Computing (T&S). T&S Computing users have too often to cope with problems that are not part of their core job. Since they have to use advanced IT resources, they need to learn and use a lot of IT low level tools (telnet, ftp, shell scripts and aliases, ...) in order to reach and use the computing resources they need. They often have also to solve very difficult security problems (access, authorization, privileges) that usually happen when the resources are not directly/locally managed. Moreover, once they get to the resources, most of the time they do not "speak" user's own language, but they still speak an Unix dialect or proprietary code languages full of command line switches, pathnames, etc. This definitely slows down the user'w own job, and might as well be not accepted, especially when technology changes very fast as it is the case in this field and new dialects need to be learned on an almost day-by-day basis. Different Operating Systems (e.g. Unix flavors and Windows flavors) raise integration issues in key areas like file-system

management, security, remote systems control, etc. thus increasing total ownership cost. EnginFrame was designed as a tool to give a Internet-ready interface to activities typically command-line oriented. Overtime it has evolved its features and capabilities, and can now be considered as a comprehensive solution to quickly and effectively building Computing Portals, i.e. intuitive web-based interfaces to computing resources. Its native language is XML, the standard that is gaining heavy backing from big names in the IT market. EnginFrame then "translates" XML into a more appropriate language depending on the client device (typically HTML, but also WML, PDF and enriched XML). This provides very high flexibility in contents presentation and users' experience, without the need of proprietary standards. The introduction of the Computing Portal concept adds one level of abstraction that allows to address both users' and system administrators' problems. Users actually enjoy an experience similar to the usual Internet surfing, browsing a Portal that actually speaks their own language and simply provides results upon requests. Help about using the services, if needed, is delivered by the portal itself in the form of HTML pages. On the other side, administrators have now more fine grained control over what users can and can not do, how allowed things are performed, and which resources are used for every service. Behind the scene any sort of technology might be used, and actually nobody will care as long as the service is up and running with the expected performance, as it commonly happens to Internet services like search engines and web-mail providers. EnginFrame was built to enable an easy and painless migration to the Computing Portal paradigm from a command-line based world, trying to re-use most of the existing methodologies with a tailored Web-like look&feel: it hides the complexity of technical computing environments behind the scenes. As new software or methodologies are implemented, the Portal can be extended in minutes to include them without the users even noticing the difference, or maybe a part of the Portal can be used as a test-bed by some users, without interfering with established methodologies. In the same way, new policies can be introduced by just changing how services are provided and/or presented, and users will use them exactly as they are supposed to do, thanks to a very high flexibility of its native language such as the XML dialect. EnginFrame addresses also the Unix/NT integration by making extensive use of the available Internet standards (HTML, HTTP, JAVA, XML, etc.), and takes care that different browsers will be properly supported.

EnginFrame makes use of the most recent mainstream standards and technologies, and integrates them in order to provide an efficient Portal Technology for the exploitation of T&S computing resources. EnginFrame foundation technologies as well as its principal features will be discussed in the following subsections.

2.1 From LAN Integration to the Computing Portal

First releases of EnginFrame were built as JAVA [10] stand-alone applications in order to provide a platform independent interface to simplify computing resource utilization in complex environments (especially well suited for mixed Unix/NT clusters).

Overtime the experience gained in the field has led to the paradigm of a Computing Portal as shown in fig. 1.

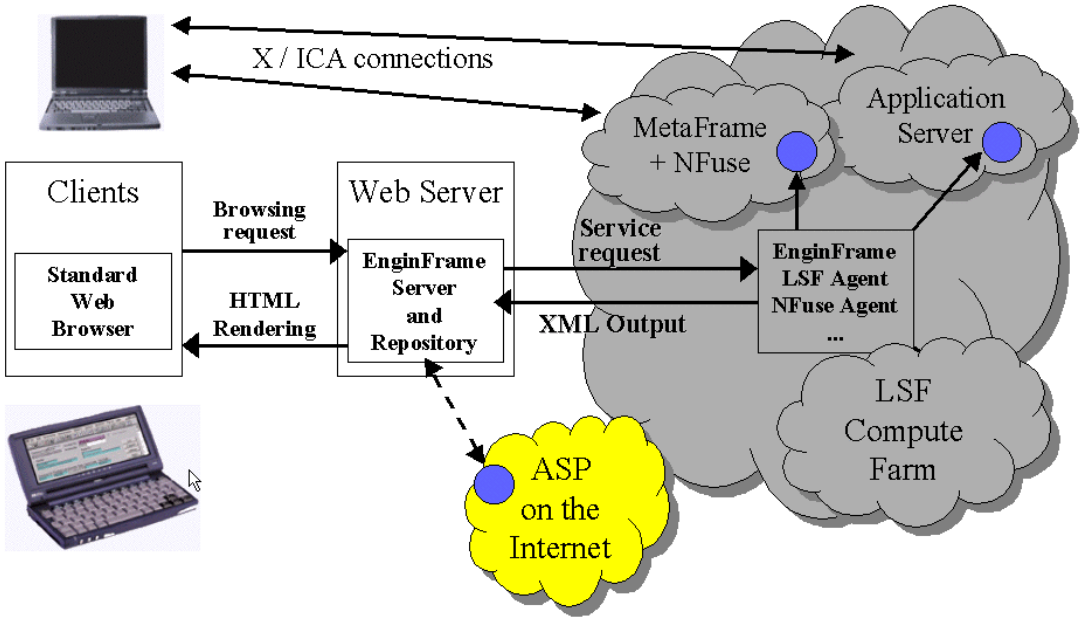

Figure 1: The Computing Portal with `EnginFrame`.

2.2 The `EnginFrame` tier model

The architecture of `EnginFrame` is logically divided into three tiers (see fig. 2):

- Client Tier, which basically consists of any browser and its extensions, and provide a comfortable framework for the users to interact with;
- Server Tier, in which one or more servlet-enabled web servers actually provide contents and services to the clients, and control resource activities in the back-end;
- Resource Tier, where a number of "Agents" control the actual computing resources (clusters, stand-alone hosts, etc.) and provide properly formatted results to the servers.

Sometimes, two tiers may actually be overlapping on the same machine (e.g., Web server or client being part of a Resource cluster). Nevertheless, the logical and practical split into three tiers gives the highest freedom. The base building block of

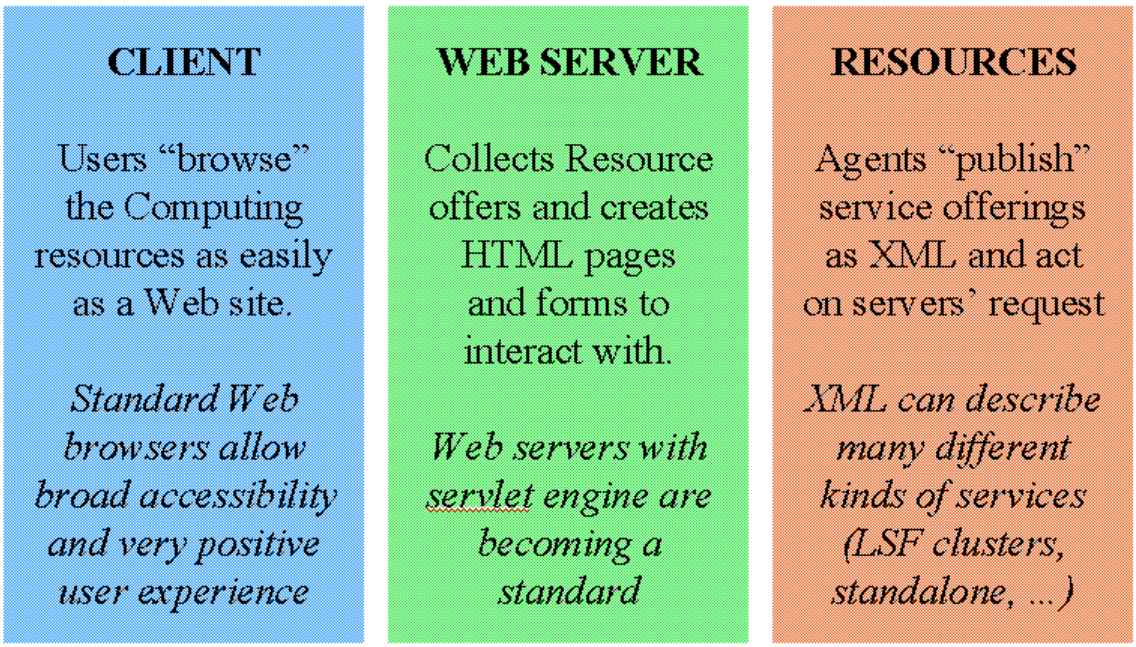

Figure 2: The `EnginFrame` three tier model.

the portal is the service, which is an `XML` representation of any computing related facility.

2.3 The Computing Portal

The typical `EnginFrame` work-flow is sketched in fig. 3. It can be compared to that relative to sending an e-mail using a Web mail service. Similarly to the web mail service, the user enters his/her own area providing credentials. If the Server Tier accepts these credentials, it presents to the user a complete web site with the available services (e.g. solvers, compilers, etc.). When a job is requested by the user, the Server selects an agent capable of providing such a service, and forwards the request to run a particular command with the data provided by the user. As a result, the Agent sends back a `XML` page describing the result, that is to be presented to the user. Inside the result there might be the actual result of the job (for very small jobs), or the acknowledgment that the job is being cared by the cluster manager, or simply by the OS of the host. This is similar to the acknowledgment that "the e-mail has been successfully sent". Hence after, the user will check his "job-box" to see if jobs have finished, or he/she be notified by e-mail for their completion, depending on site policies. Results will then be delivered as described further on.

Depending on how "fat" the client is, and which features the browser supports, both the kind of service and the format of output that can be delivered to the user

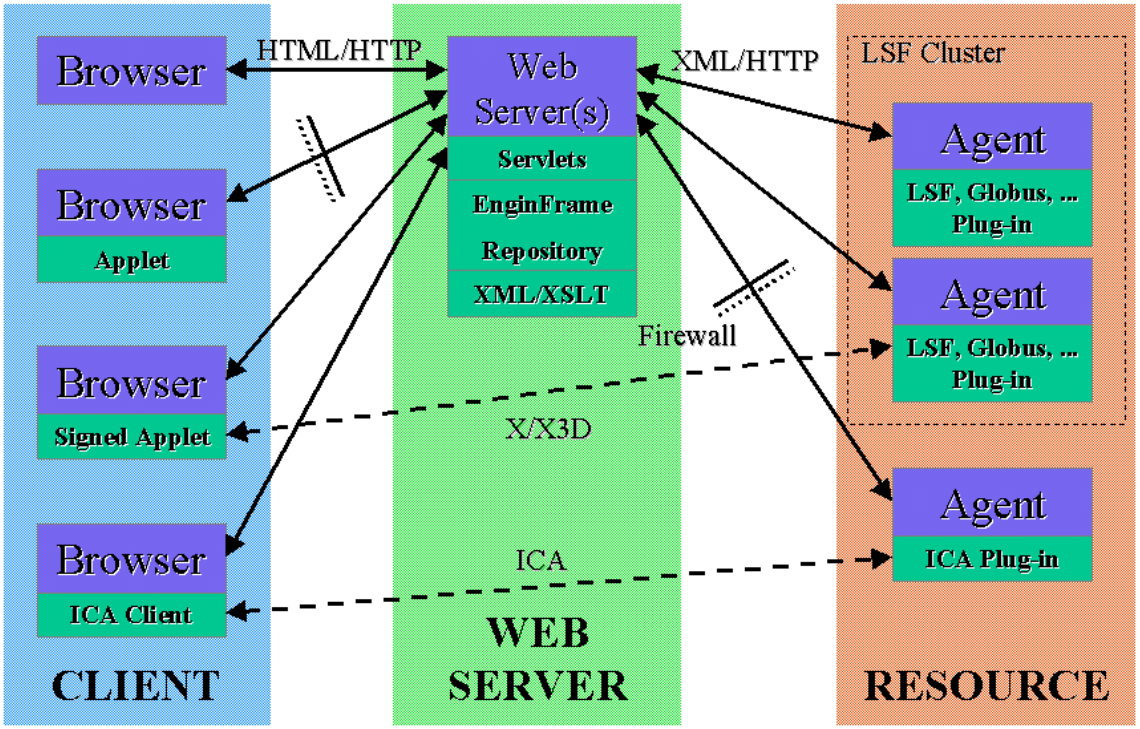

Figure 3: The `EnginFrame` work-flow.

may widely vary. Features like Remote File Browsing or graphical output of remote Windows applications need more capable browsers/clients than simpler services. As most people in the T&S computing have experienced, an `HTML` browser might not provide by itself enough capabilities to handle the user's real needs. For this reason, one of the key strengths of `EnginFrame` is integration. As a matter of fact, services are often provided complementing the Web with third party technologies in a way that is transparent to the user. The Resource Tier as well uses a plug-in mechanism to provide best integration with underlying computing resources. Currently an LSF [8] plug-in is in its final stage, and the `Globus` one is in its way to be developed. Fire-walls often restrict the possibilities of users. For this reason, `EnginFrame` provides several options that address Firewall-aware communication issues only when and if needed [7].

2.4 `EnginFrame` services

The design goals of `EnginFrame` are:

- simplicity, no need to be too verbose when you don't need to;
- easy prototyping, so that building a new service is a matter of minutes;

- effectiveness, to properly address the critical issues in computing resources;
- adherence to standards, in particular, to the Apache Group implementations;

Through this language, that can be edited with any XML editor of your choice, or automatically built by step-by-step wizards, it is possible to describe services provided by the Agents. The service description includes the name of the service, the options we have to specify, the command that has to be executed, and, optionally, other information for the inexperienced user. Everything else is generated dynamically, without the need of further coding. Unless you need to change the overall look&feel of the Portal, which is anyway possible with very little effort, the services can actually be published in minutes without even knowing a single HTML tag.

2.5 Requirements for Server

In conclusion, for the sake of completeness, the following is the list of packages needed/used by EnginFrame [11]:

- Apache Web Server 1.3.12-2 or higher;
- Apache JServ 1.1.2-1: the original JAVA Servlet module for Apache;
- Xerces-j 1.1.2-1: Xerces-j is an DOM 2 and SAX 2 compliant XML parser. It is used by Xalan, FOP, and Cocoon;
- Xalan-j 1.1.D01-1: Xalan-j implements the current W3C recommendations for XSLT and the XML XPath language. It requires Xerces-j;
- FOP 0.13.0-1: FOP (Formatting Objects Processor) implements the W3C recommendations for XSL:FO, generating a PDF document;
- Cocoon 1.7.4-1 or higher: Cocoon is a framework that allows an XML document to undergo a series of transformations. The most obvious application allows a web server to transform XML content into HTML output;
- JDK 1.2.2 or higher: the JAVA Development Kit by SUN Microsystems.

More extensive information about EnginFrame can always be found at the URL: http://www.enginframe.com

3 Current implementation

The current implementation of the computing portal EnginFrame, including the ALICE specific services discussed in the following, is accessible from the URL: http://gridct1.ct.infn.it/globus.

Figure 4 shows the home page of the web interface. In the rest of this note

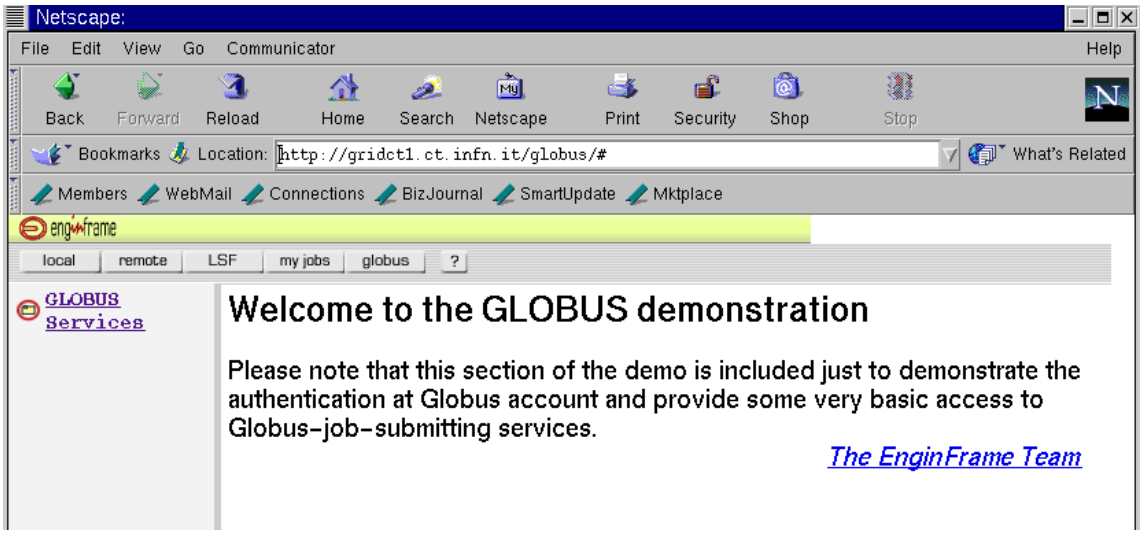

Figure 4: The main screen of `EnginFrame`.

we will only describe the "`Globus` Services". Clicking with the mouse on "`Globus` Services", one gets the list of the currently available services as it is shown in fig. 5. All the services are described in detail in the following subsections.

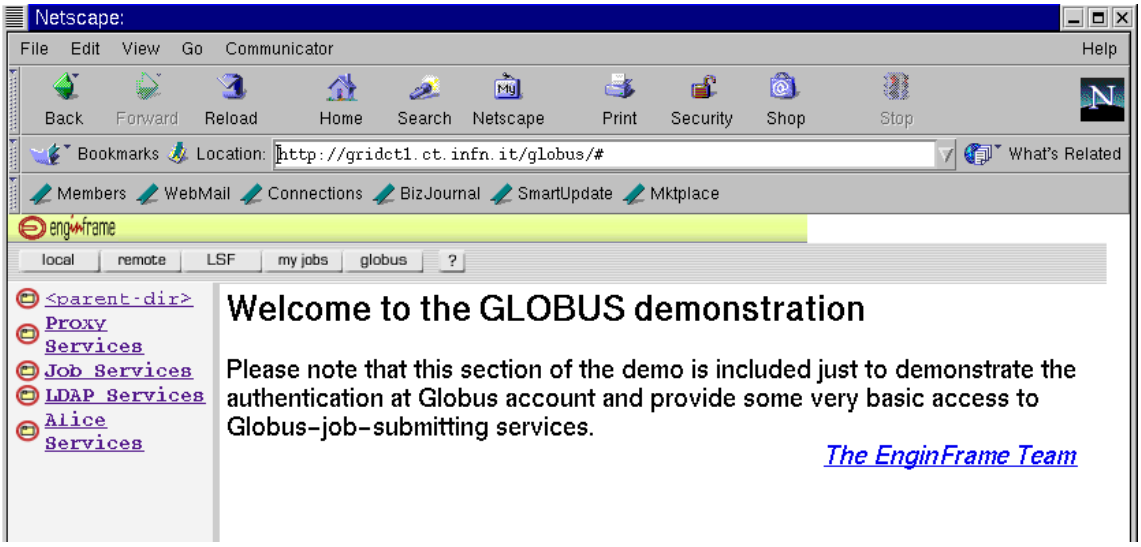

Figure 5: The list of available `Globus` services.

3.1 Proxy services

"Proxy services" allow the user to set/check/cancel his/her authentication to the "GRID" using his/her own digital certificate. As described in fig. 6 there are

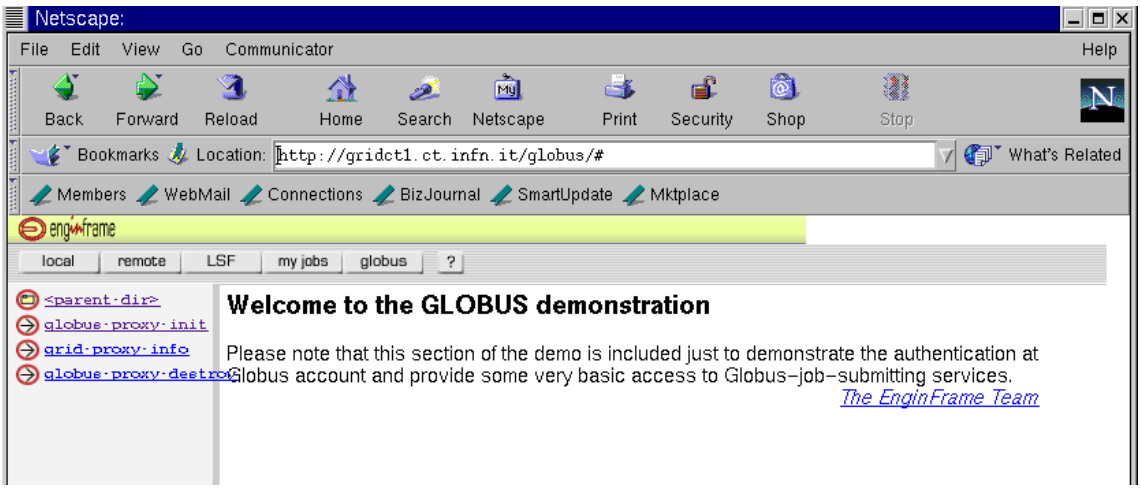

Figure 6: The list of available Proxy services.

three "Proxy services" available: i) `GRID-proxy-init`, ii) `GRID-proxy-info`, and iii) `GRID-proxy-destroy`. They are completely equivalent to the corresponding command line `Globus` commands.

Mouse clicking on the first one, the user is prompted to introduce both his/her name and PEM phrase (password) of his/her certificate as shown in fig. 7. The password is then passed to the GSI service of `Globus` for verification. If it is correct, then the user gets a permit to run on the "GRID" machines the world over where his/her certificate is mapped onto a valid/existing local user for a finite time of 12 hours (this is the default time of the corresponding `Globus` command). The information about the public certificate, its type, strength and the time left is returned on the screen. However, for security reasons, if he/she closes his/her browser (even before the 12 hours time limit) the authentication token is destroyed and he/she must login again when a new browser is opened.

After a successful login the user can check the time left before the expiration of his/her token clicking on `GRID-proxy-info`. The output returned by this service is shown in fig. 8. The third service, `GRID-proxy-destroy` eliminates the authentication token still keeping the browser open. For security reasons, the use of this service is recommended if the user accesses the "GRID" from a machine shared with other people. As explained before, the same effect can be reached, however, killing the browser.

For obvious reasons, `GRID-proxy-init` should be the first `EnginFrame` service used by the user. However, there are checks throughout the portal that prevent the

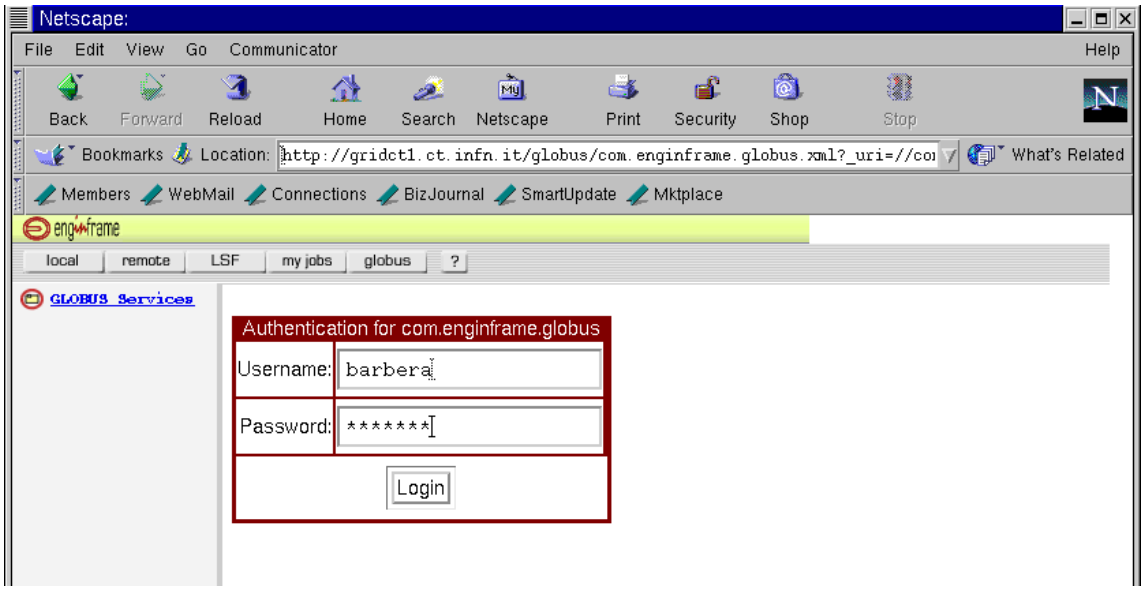

Figure 7: The `GRID-proxy-init` service with the authentication window.

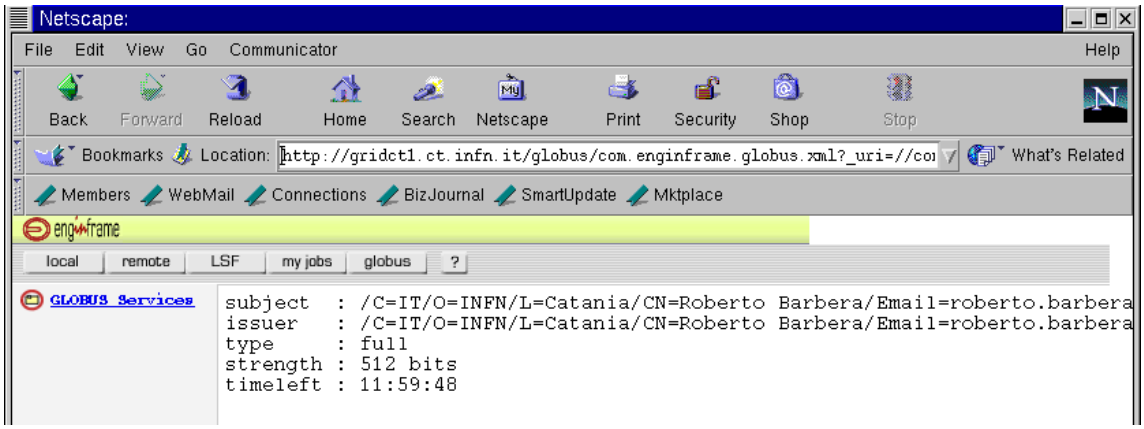

Figure 8: The `GRID-proxy-info` service with its output.

use of any service if the user did not get the authorization token.

3.2 Job services

"Job services" allow the user to launch/check jobs on the "GRID" machines the world over where his/her certificate is mapped onto a valid/existing local user. As shown in fig. 9, there are four "Job services" currently implemented: i) `globusrun`, ii) `globus-job-run`, iii) `globus-job-submit` and iv) "Job Status".

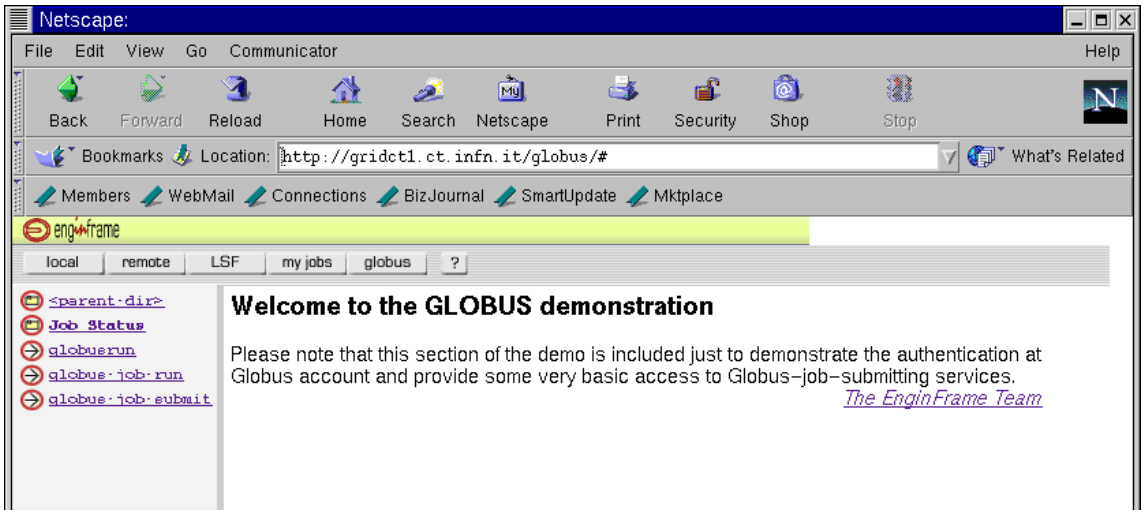

Figure 9: The list of available Job services.

The `globusrun` service is the most general way to submit interactive jobs (interactive means here that `EnginFrame` gets blocked until the job is completed), even if it not the easiest one. Its use is based on the `Globus` Resource Specification Language (RSL) [9]. The job submission form of `globusrun` is shown in fig. 10. The user has to provide `EnginFrame` with the name of the remote "GRID" machine he/she wants to run on and the RSL file containing the job description. The output of the job, if any, is returned on the browser page after its completion.

The `globus-job-run` service is another way to launch interactice jobs on the "GRID". The job submission form of `globus-job-run` is shown in fig. 11. The user has to provide `EnginFrame` with the name of the remote "GRID" machine he/she wants to run on and the OS commands or shell script to be executed on the remote machine. The output of the command/script, if any, is returned on the browser page after job completion.

The `globus-job-submit` service is the way to launch batch jobs on the "GRID" (batch means here that `EnginFrame` spawns the job keeping trace of the job ID returned by `Globus`). The job submission form of `globus-job-submit` is shown in fig. 12. The user has to provide `EnginFrame` with the following information:

- a job description;
- the name of the "GRID" machine he/she wants to run on;
- the job-manager to run the job with on the remote machine;
- the path/name of file containing the standard output log which will reside on the disk of the submitting machine;

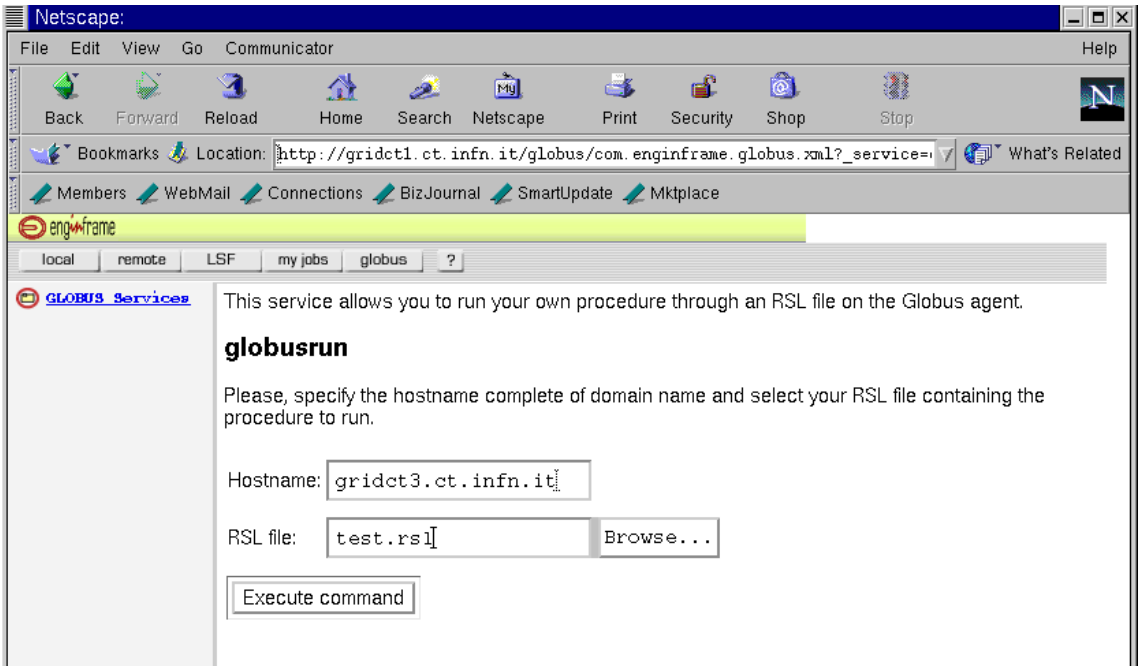

Figure 10: The `globusrun` service with the job submission form.

- the path/name of file containing the standard error log which will reside on the disk of the submitting machine;
- the OS command or shell script to be executed on the remote machine.

After submission, `EnginFrame` will pass and keep trace of the job ID returned by `Globus` which is necessary to monitor the job status.

"Job status services" will be described in Section 4, in correspondence with the "ALICE services".

3.3 LDAP services

"LDAP services" are intended to interface `EnginFrame` with the information system of the "GRID" based on the LDAP protocol through the GIS service of `Globus`. As shown in fig. 13 two "LDAP services" have been implemented so far: i) "People" and ii) "Resources". They are described in detail in the next two subsections.

3.3.1 People

The "People" service allows the user to connect to and browse/edit a LDAP server. In order to do that `EnginFrame` has been interfaced with LDAP Browser/ Editor [12]. This is a program written in `JAVA` which can be run both in a stand-alone mode

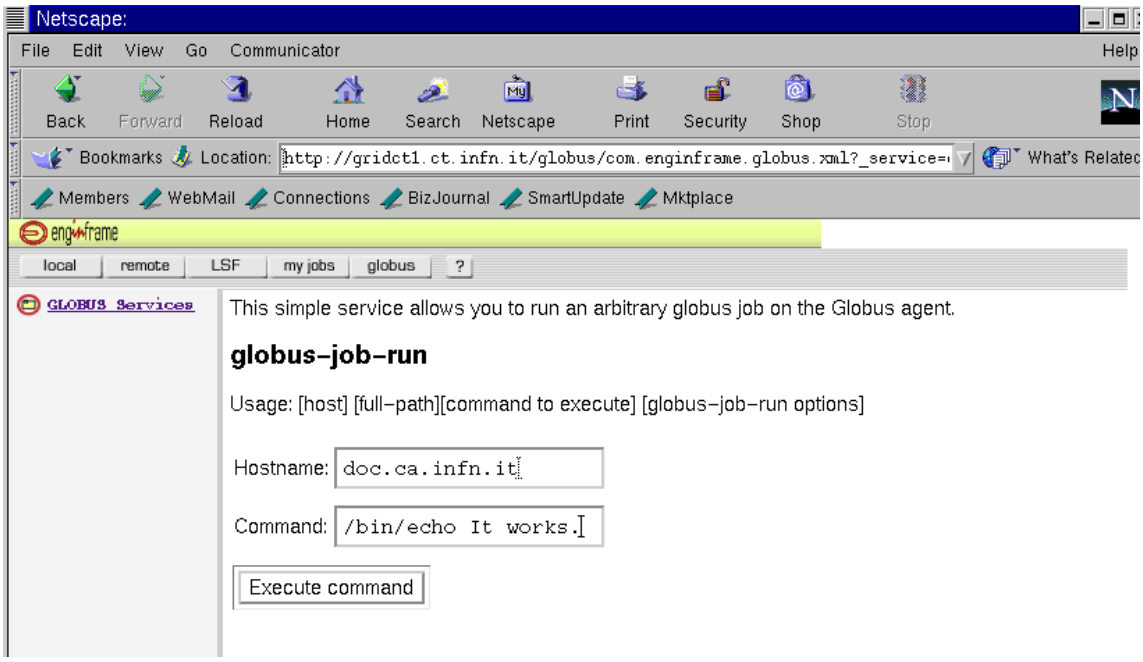

Figure 11: The `globus-job-run` service with the job submission form.

and inside a web browser (this is the way it works with `EnginFrame`). Up to now, the link has been set only with the INFN LDAP server [13] but other servers can be added very easily as they will become available.

When he/she starts the "People" service by clicking on it, the user is prompted for the name of the server and some connection options, as shown in fig. 14 At this moment it is possible to choose between the default read-only connection (just to browse the server) and a read-write connection (which is the case shown in the figure) where the user can add/modify/delete the fields of the database remotely. It is worth noting here that LDAP Browser/Editor supports secure connections via the SSL protocol.

Once the connection has been established, the user can browse/edit the database as shown in figs. 15-16. In the INFN LDAP server record are sorted both by site (useful when one wants to have geographic map of your test-bed) and experiment (useful when one wants to have access to people belonging to a given experiment). Figure 15 shows an example of the first case while fig. 16 shows an example of the second. In this latter case, it is possible to see from the figure how people can be subdivided among experiment's sub-parts and/or tasks if one wants to map different "GRID" users to different accounts on the local farms having different running priorities. In fact, a completely automatic tool called `certretrieve` [14], included in the INFN GRID Distribution Toolkit [15], can access the LDAP server, download the certificates and populate the local `GRID-mapfile`'s[9].

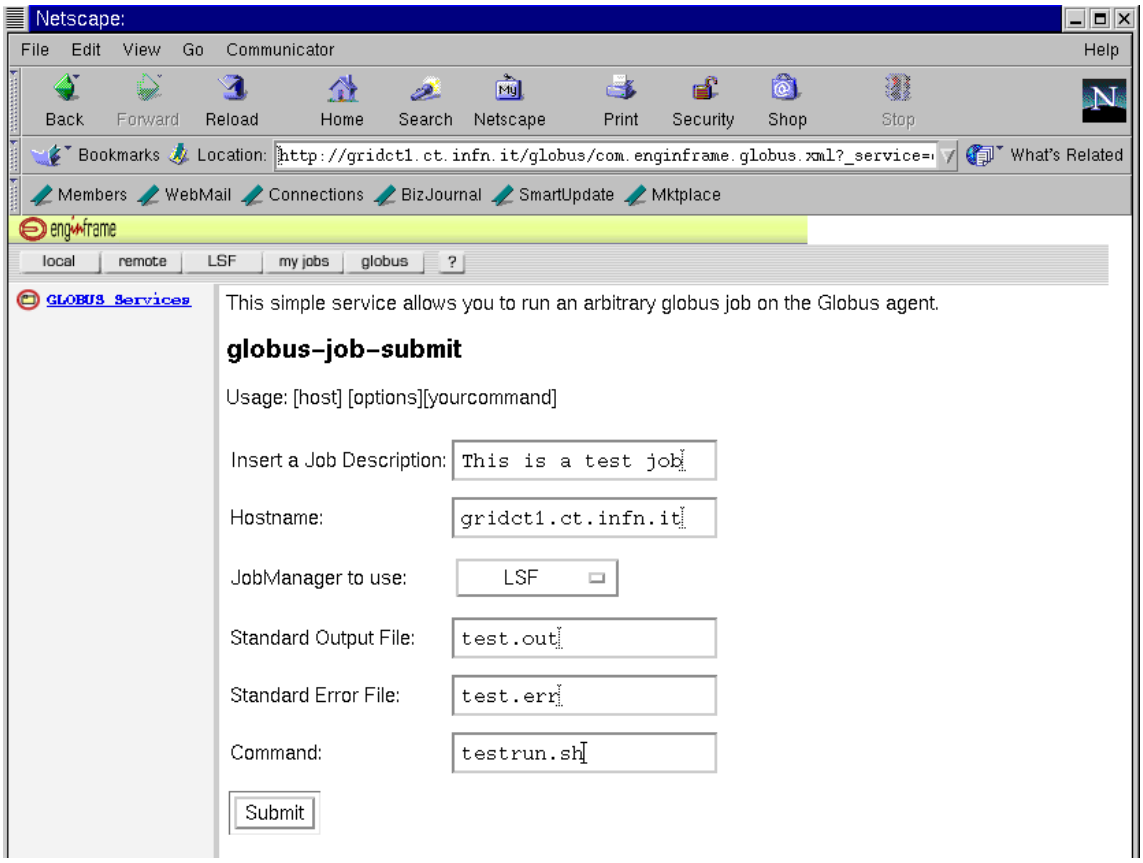

Figure 12: The `globus-job-submit` service with the job submission form.

3.3.2 Resources

The "Resources" service allows the user to connect to a LDAP server to get information about the computing resources available. Up to now, `EnginFrame` has been interfaced only with the INFN Metacomputing Directory Service (MDS) server [16] but other servers can be added very easily as they will become available.

When he/she starts the "Resources" service by clicking on it, the user gets the list of available results sorted by site, as shown in fig. 17. It is possible to browse the database to get information on a single "GRID" node (see fig 18), on its hardware configuration (see fig 19), and even on its batch system queues and `Globus` installation (see fig 20).

4 The ALICE services

All `EnginFrame` services described so far are "general purpose" and can be used by any authorized "GRID" user. In order, from one side, to test the computing

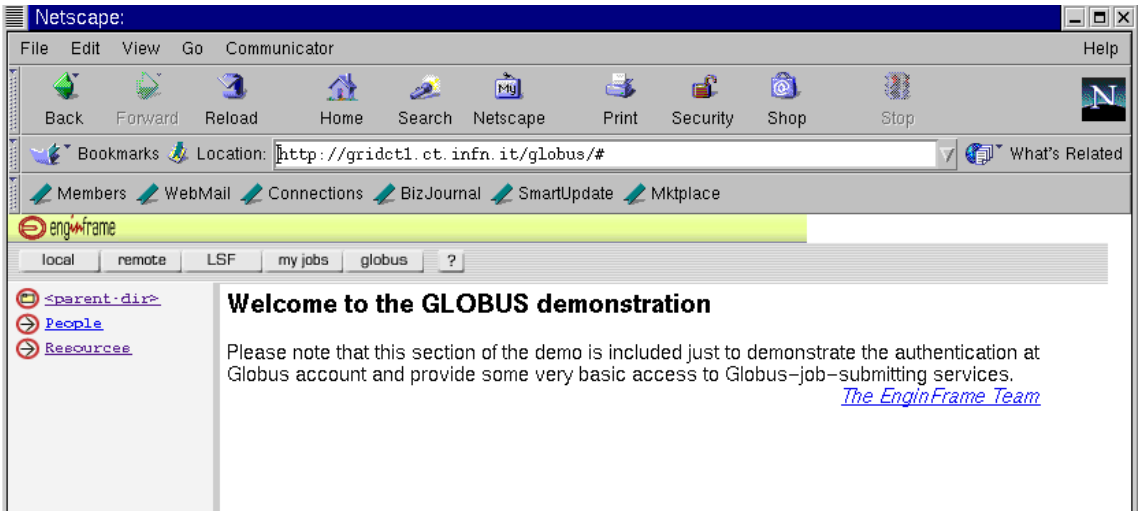

Figure 13: The list of available LDAP services.

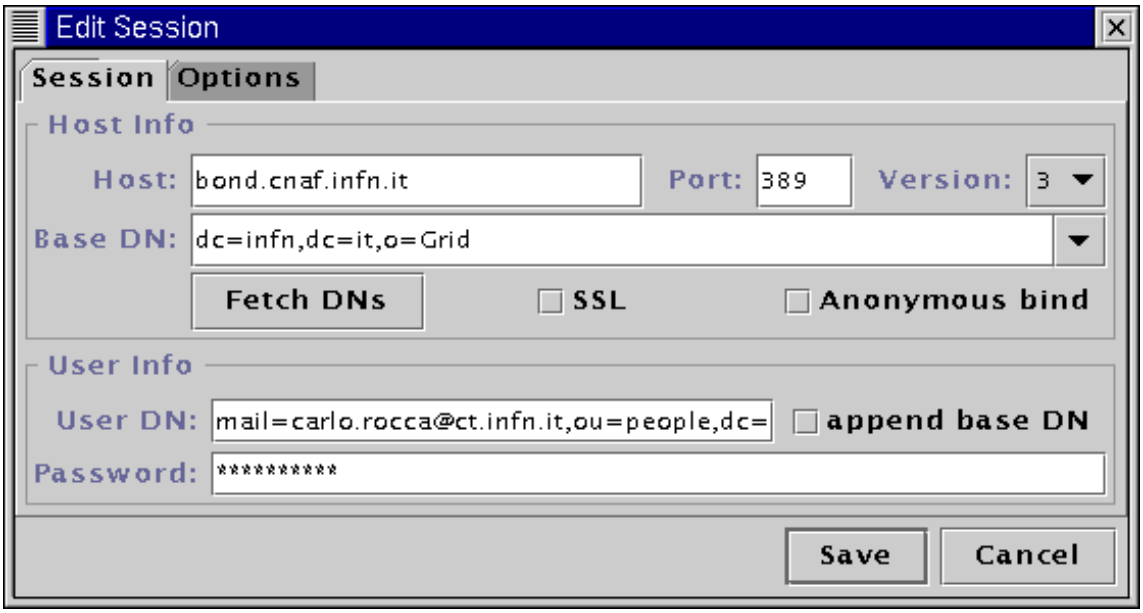

Figure 14: The parameters of the INFN LDAP server as defined in the "connection window" of LDAP Browser/Editor.

portal with a "real life" use case in the HEP community and, on the other side, to establish something usable for the MonteCarlo production foreseen in ALICE in 2001 for the completion of the Physics Performances Report (PPR), we have customized special services of `EnginFrame` for the ALICE needs. This would not have been

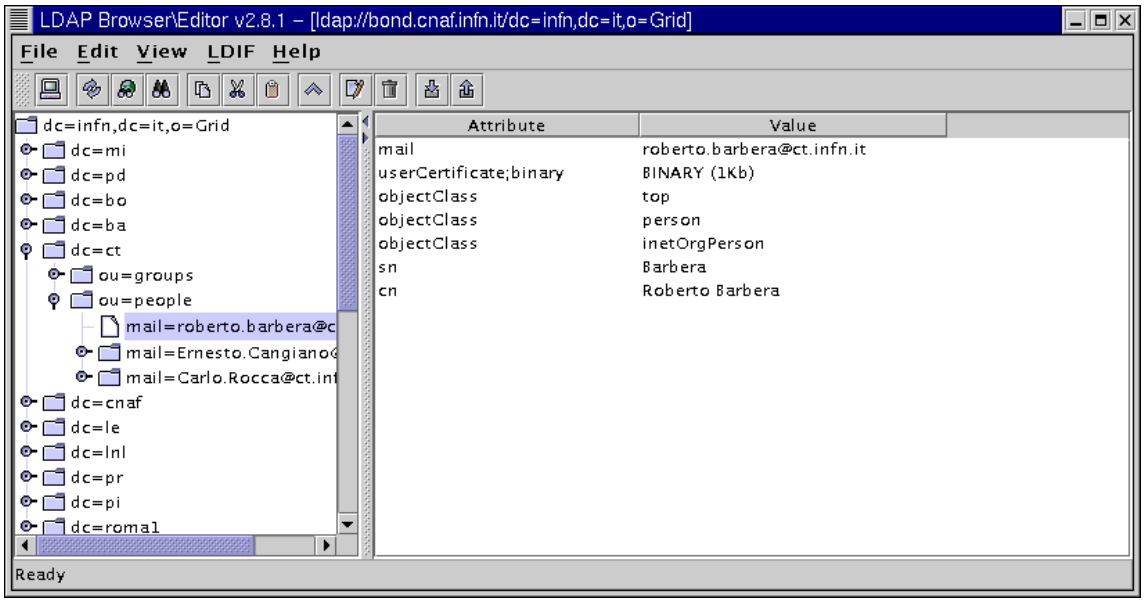

Figure 15: A "site-sorted" browsing of the INFN LDAP server.

possible without the precious experience gained within the ALICE Collaboration in the last few months and the results of the tests of remote distribution and distributed production with the available "GRID" services made between Cagliari, Catania, CCIN2P3 at Lyon, Ohio Supercomputing Center, and Torino. The results of these tests will be widely reported in another forthcoming ALICE Internal Note.

Two "ALICE services" have been implemented so far: i) "Submit Job" and ii) "Monitoring". They are described in detail in the next two subsections.

4.1 Submit Job

The "Submit job" service allows an ALICE user to start a run of `AliRoot` [18] on a remote machine connected to the "GRID" and collect both the standard output and error log files on the disk of the submitting machine. The file containing the simulated event is, instead, stored on the disk of the remote machine. The job submission form of "Submit job" is shown in fig. 21. The user has to provide `EnginFrame` with the following information:

- a description of the `AliRoot` job;
- the name of the "GRID" machine he/she wants to run on;
- the job-manager to run the job with on the remote machine;
- the name of the `Root` file where the simulated event will be stored on the disk of the remote machine;

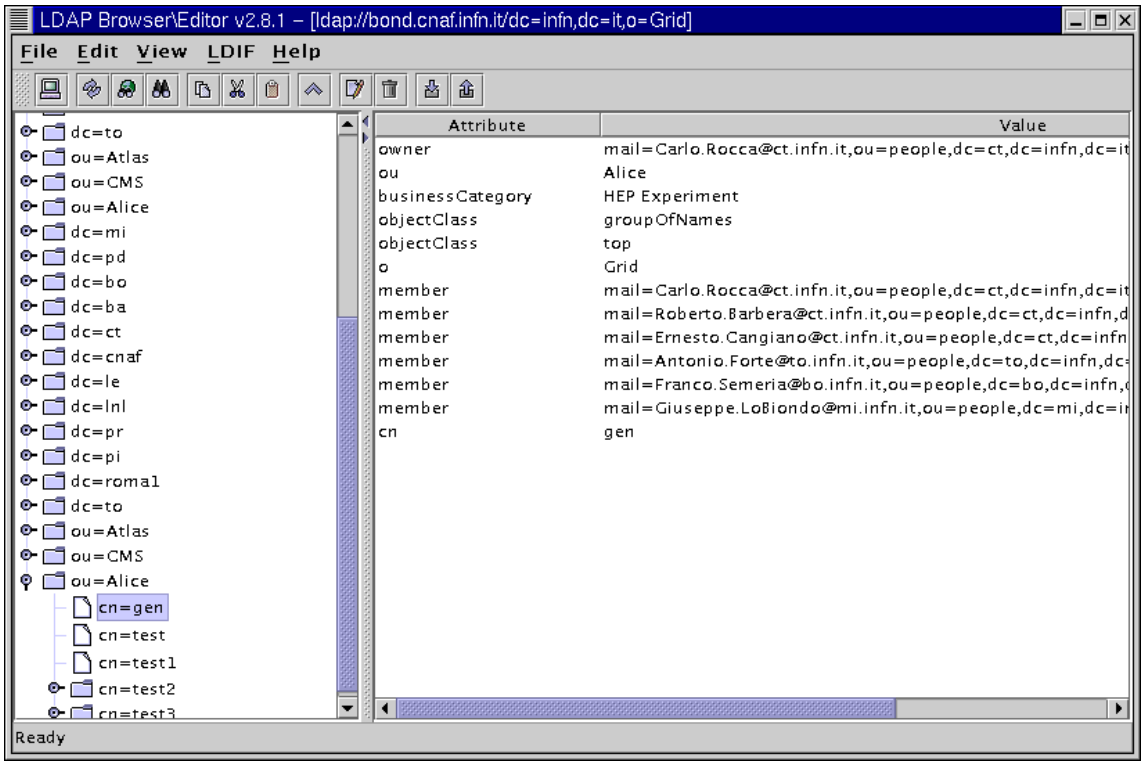

Figure 16: An "experiment-sorted" browsing of the INFN LDAP server.

- the path/name of file containing the standard output log which will reside on the disk of the submitting machine;
- the path/name of file containing the standard error log which will reside on the disk of the submitting machine;

Unlike the case of the `globus-job-submit` service discussed above, here it is not necessary to insert the name of the script needed to run `AliRoot` on the remote machine. This is because `EnginFrame` is already compatible with the environment settings included in the distribution kit of `AliRoot` distributed to the ALICE test-bed sites involved in the PPR production.

After submission, `EnginFrame` takes care and keeps trace of the job ID returned by `Globus` which is necessary to monitor the job status. The list of the currently available "Job status services" is shown in fig. 22. They are: i) "List My Jobs", ii) "View Last Job Status", and iii) "View One Job Status". The first service shows all jobs submitted by the user, as shown in fig. 23. The second service shows the status of the last submitted job, only, while the last service shows the status of a particular job. In this case, the user has to provide `EnginFrame` with the job ID returned by `Globus` for that particular job. Although they have been shown within

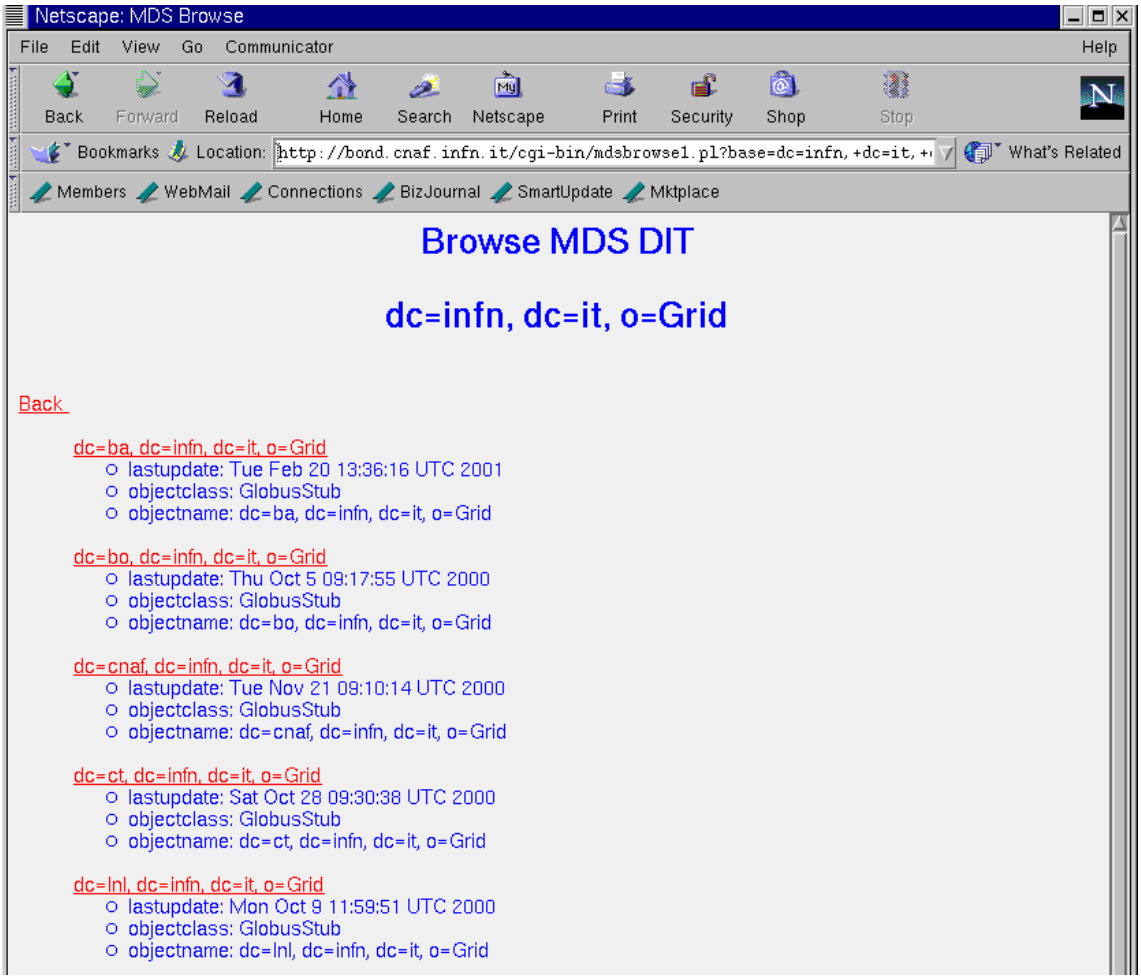

Figure 17: The INFN MDS server home page.

the context of the "ALICE services", the "Job status" services are general and can be used for all kinds of jobs submitted to the "GRID".

4.2 Monitoring

Once a job has been submitted, or before submitting a job, also, it is extremely useful to have a tool to monitor the remote farm/machine where the job is actually running.

In the first case, one can continuously check:

- if the job is running;
- since how much time;

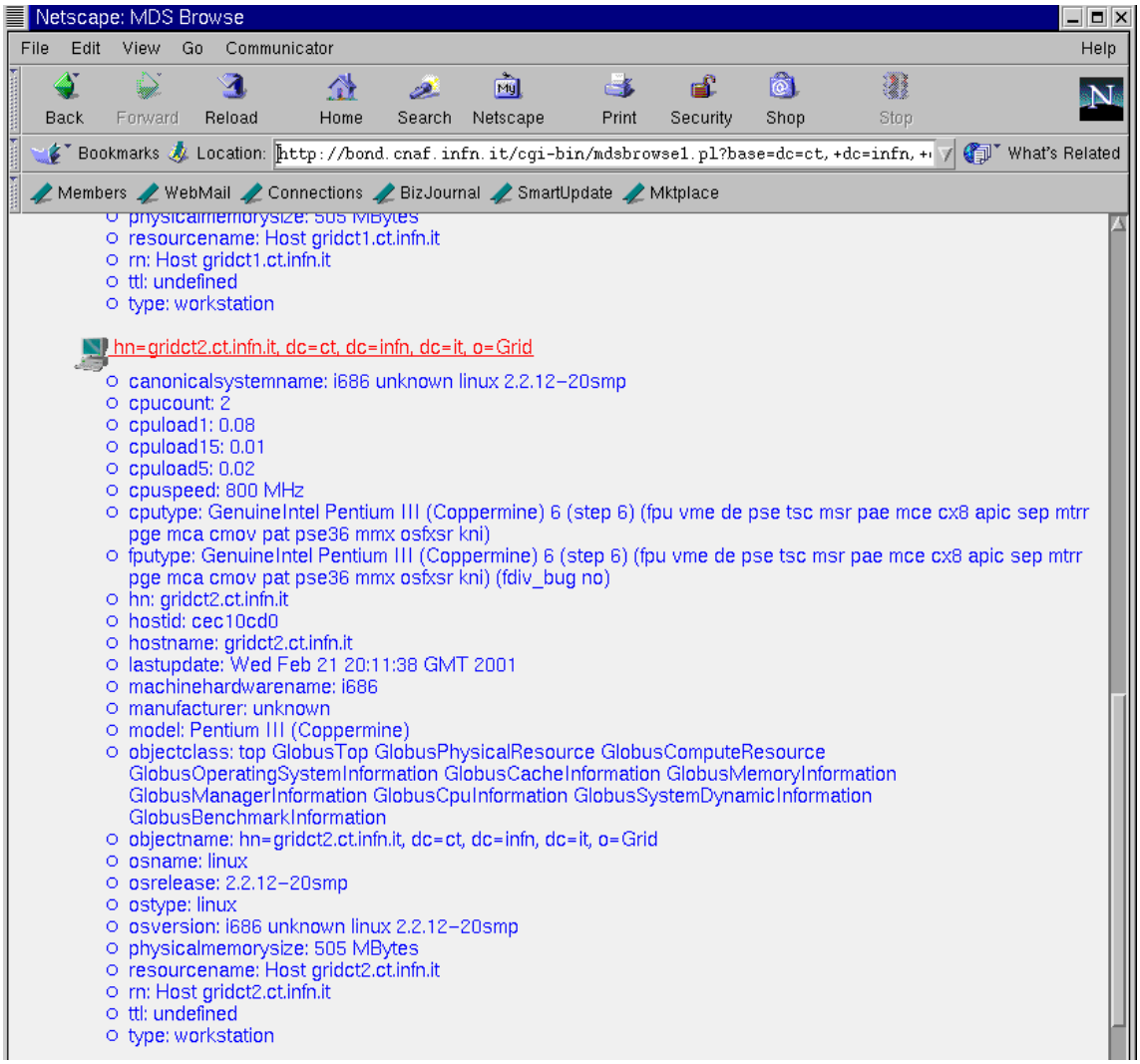

Figure 18: An INFN MDS server page with the general information about a "GRID" node.

- the CPU load of the remote machine during this amount of time;
- the disk space on the remote machine where the output file is being stored;
- the connectivity and the level of network traffic, if any, between the remote machine and the submitting machine.

In principle, this is against the "GRID" principles claiming that the users do not have to know where on the "GRID" their jobs are running, but it is an invaluable benefit when one has to run/coordinate an event production.

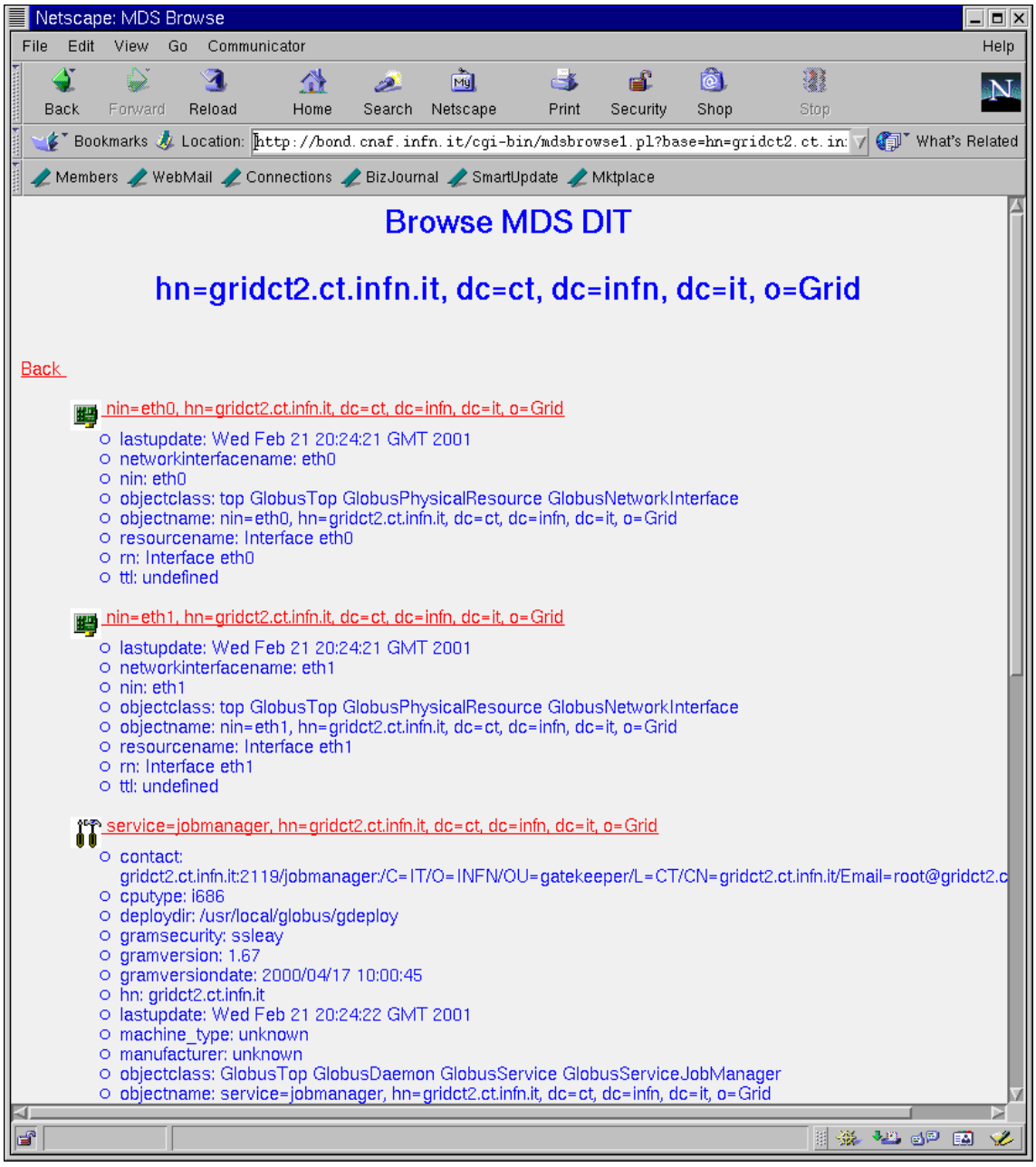

Figure 19: An INFN MDS server page with the hardware and job-manager information.

In the second case, the knowledge of the status of a remote farm/machine (CPU load and disk space) and its accessibility (network speed), can help a lot in deciding how to share the production among the computing resources available.

The "Monitoring" service interfaces `EnginFrame` with the monitoring systems of

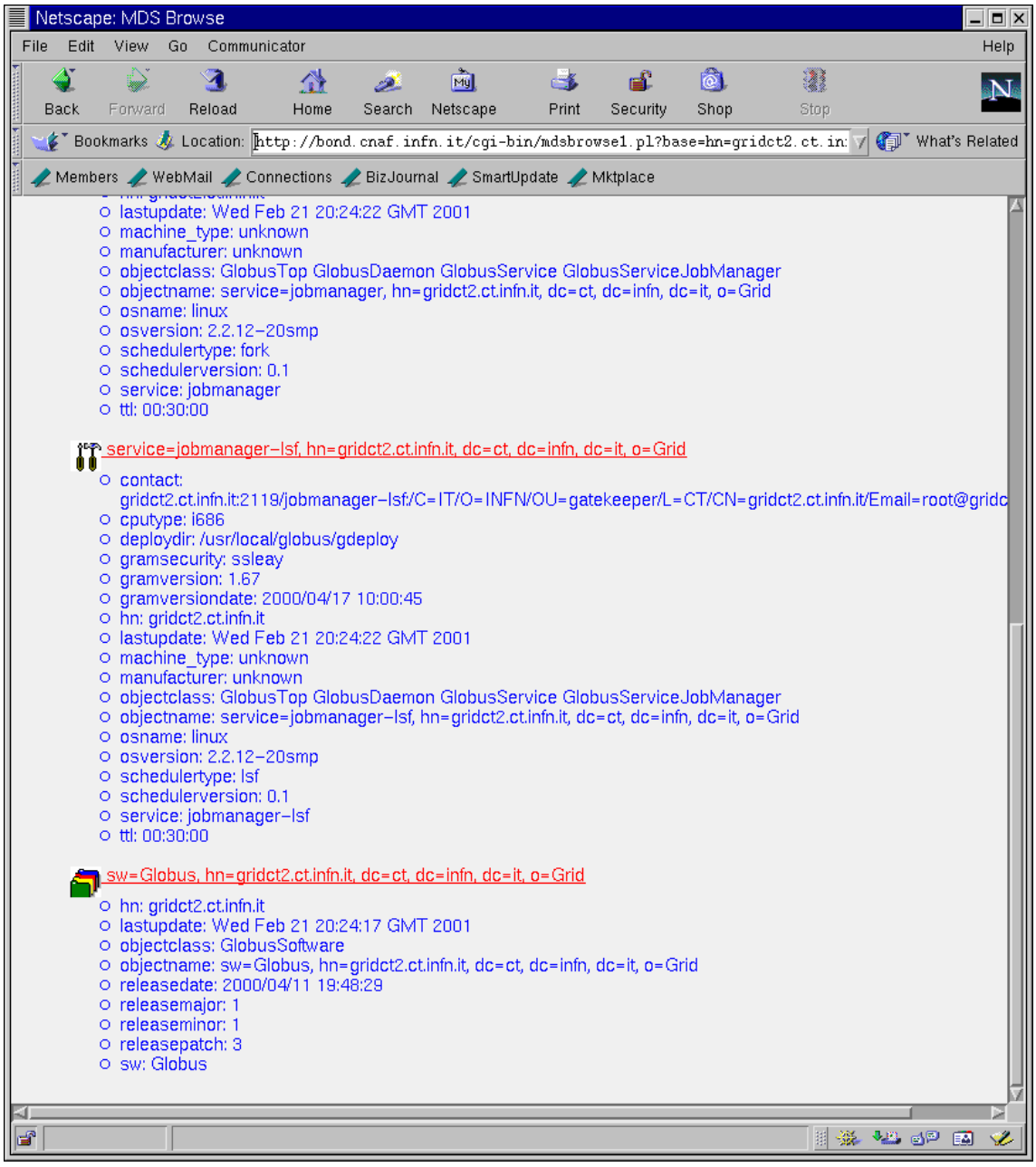

Figure 20: An INFN MDS server page with the `Globus` installation information.

the farms which are becoming available within the ALICE test-bed built on purpose for the PPR. The monitoring system adopted so far within the ALICE test-bed is based on MRTG [17] and all details will be discussed in a dedicated forthcoming note. It is worth emphasizing here, however, that this monitoring system provides a web output and seamlessly integrates with `EnginFrame` The layout of the "Monitoring"

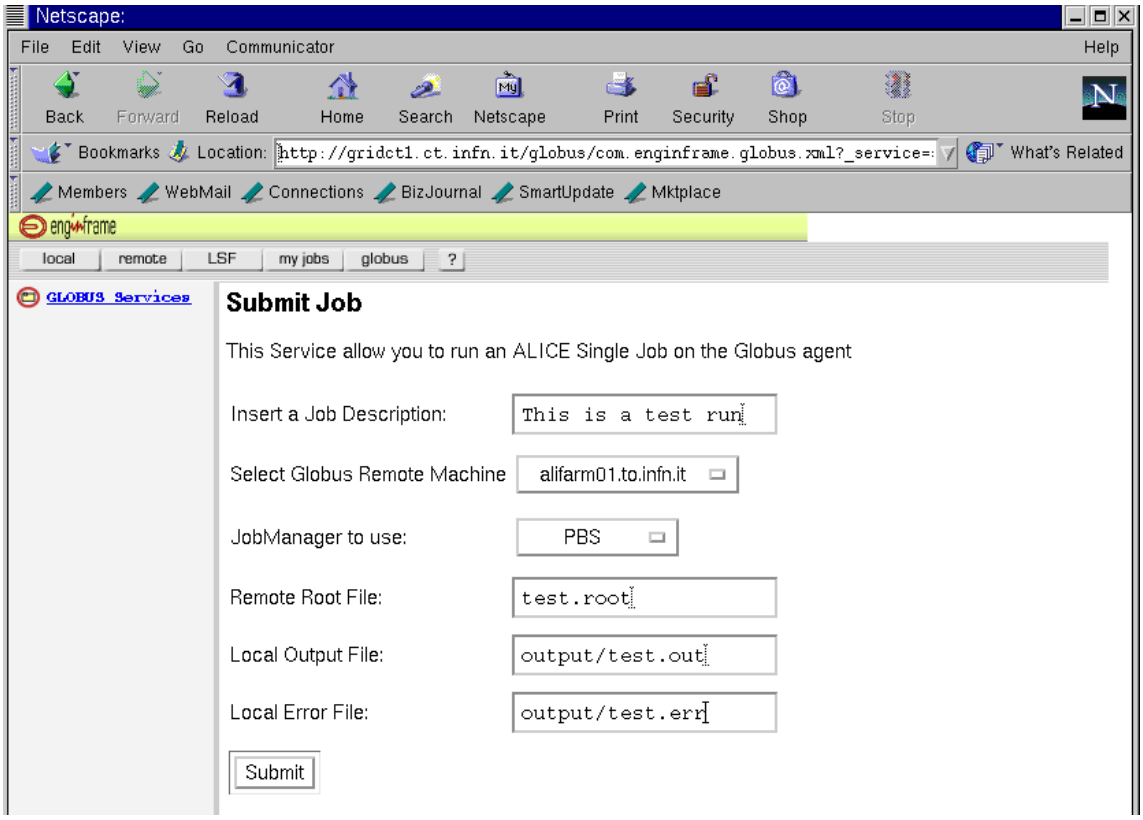

Figure 21: The Submit Job page of `EnginFrame` for the ALICE services.

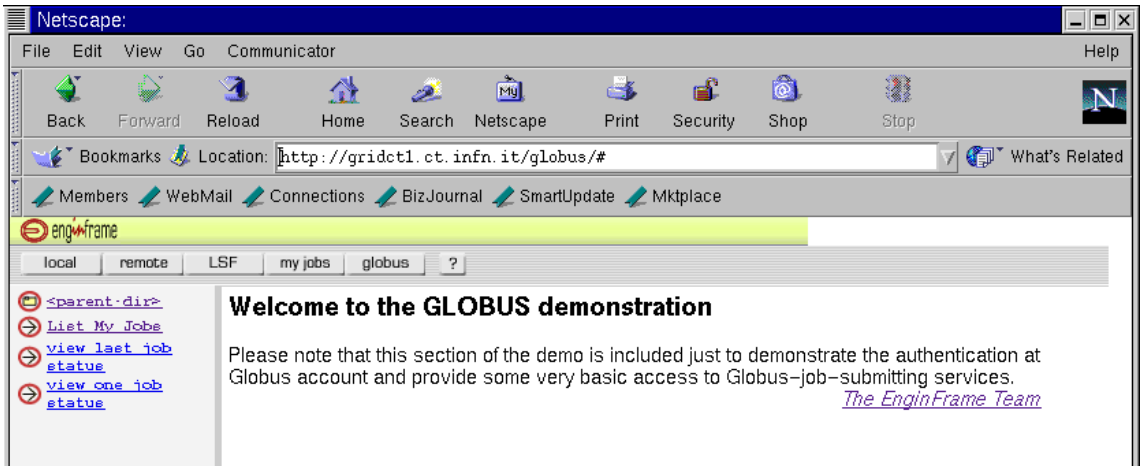

Figure 22: The list of available Job Status services.

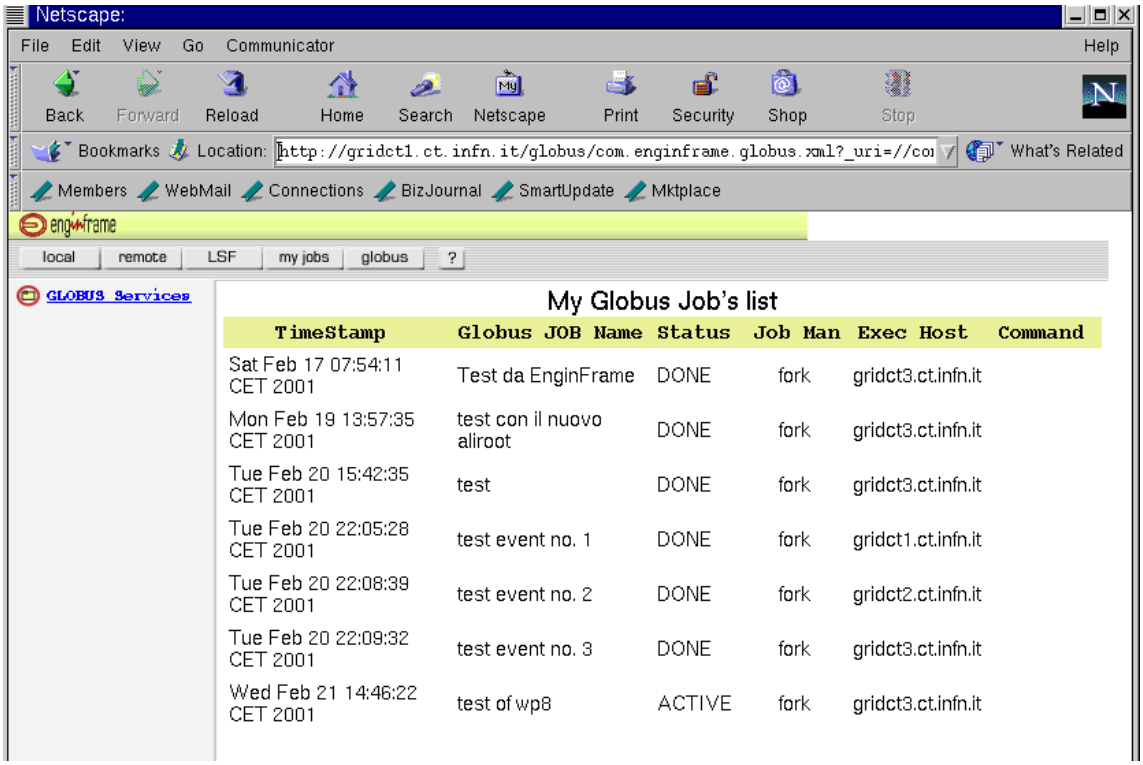

Figure 23: The output of the service "List My Jobs".

service is shown in fig. 24. For all the sites currently available, both the farm and the network are monitored. As examples, figs. 25-27 show the 'daily' and 'weekly' graphs provided by the monitoring systems of the Catania and Cagliari ALICE farms for the CPU load, the disk space occupancy, and the network latency.

5 Summary and conclusions

Although very complete and general-purpose, `EnginFrame` is far from being a "frozen" product. Future developments foreseen in the next few days/weeks will include, among the others:

- the possibility to kill a job;
- the possibility to show "on-flight" (i.e., before the end of the job) both the standard output and the standard error log files from within the web browser;
- the possibility to move the output file from the remote machine back to the submitting machine (using the `Globus` Access to Secondary Storage (GASS) service) in order to test the new "GRID flavored" FTP's;

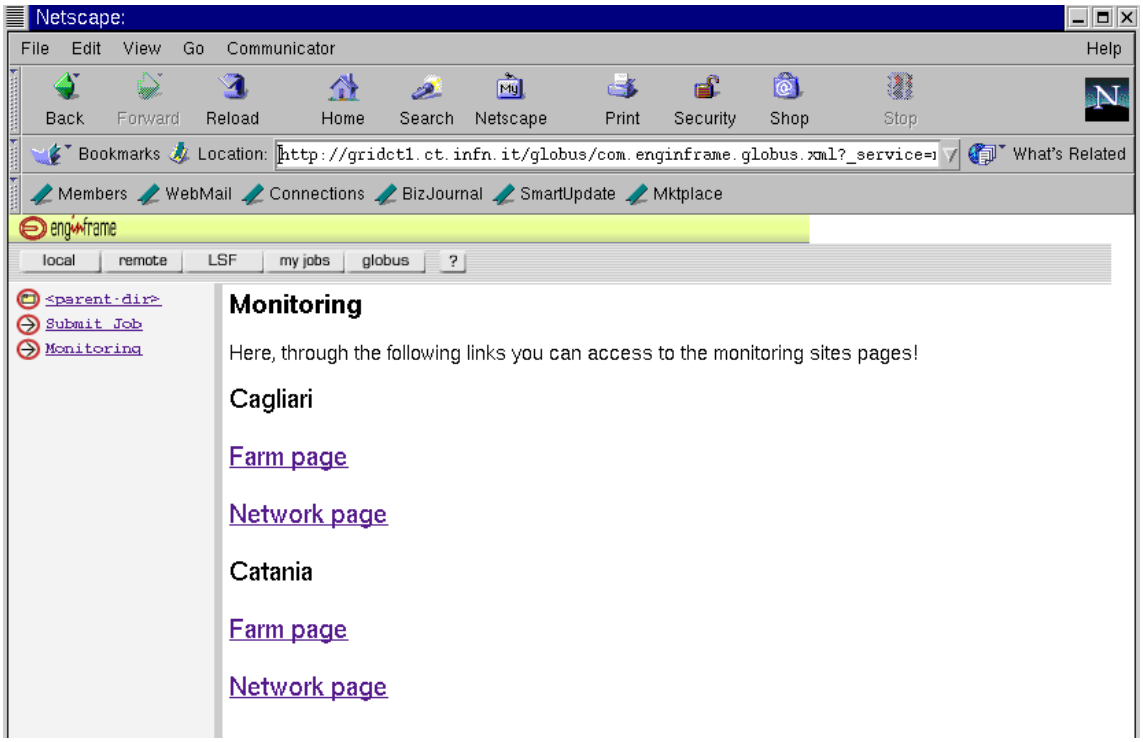

Figure 24: The Monitoring page of `EnginFrame` for the ALICE services.

- the possibility to submit m `AliRoot` jobs on n "GRID" machines at the same time keeping track of all job ID's, log files, and output files;
- a better integration, via specific and automatically generated links, between the "Job status services" from one side and the "LDAP services" and the "Monitoring services" from the other side.

However, even at this stage, it already provides the user with a robust web interface to the currently available "GRID" services and contains all the technology to seamlessly incorporate all the new ones that will be created inside the DataGRID Project in the next months/years. Due to its modularity, experiment's specific services can also be rapidly defined and added, so there is hope in the authors that other experiments/projects besides ALICE could test it in "real-life" applications and provide invaluable feed-backs.

Acknowledgments

The `GLOBUS` evaluation team inside the INFN GRID Project and the Catania IT service are also greatly acknowledged for their help in installing e configuring `Globus`.

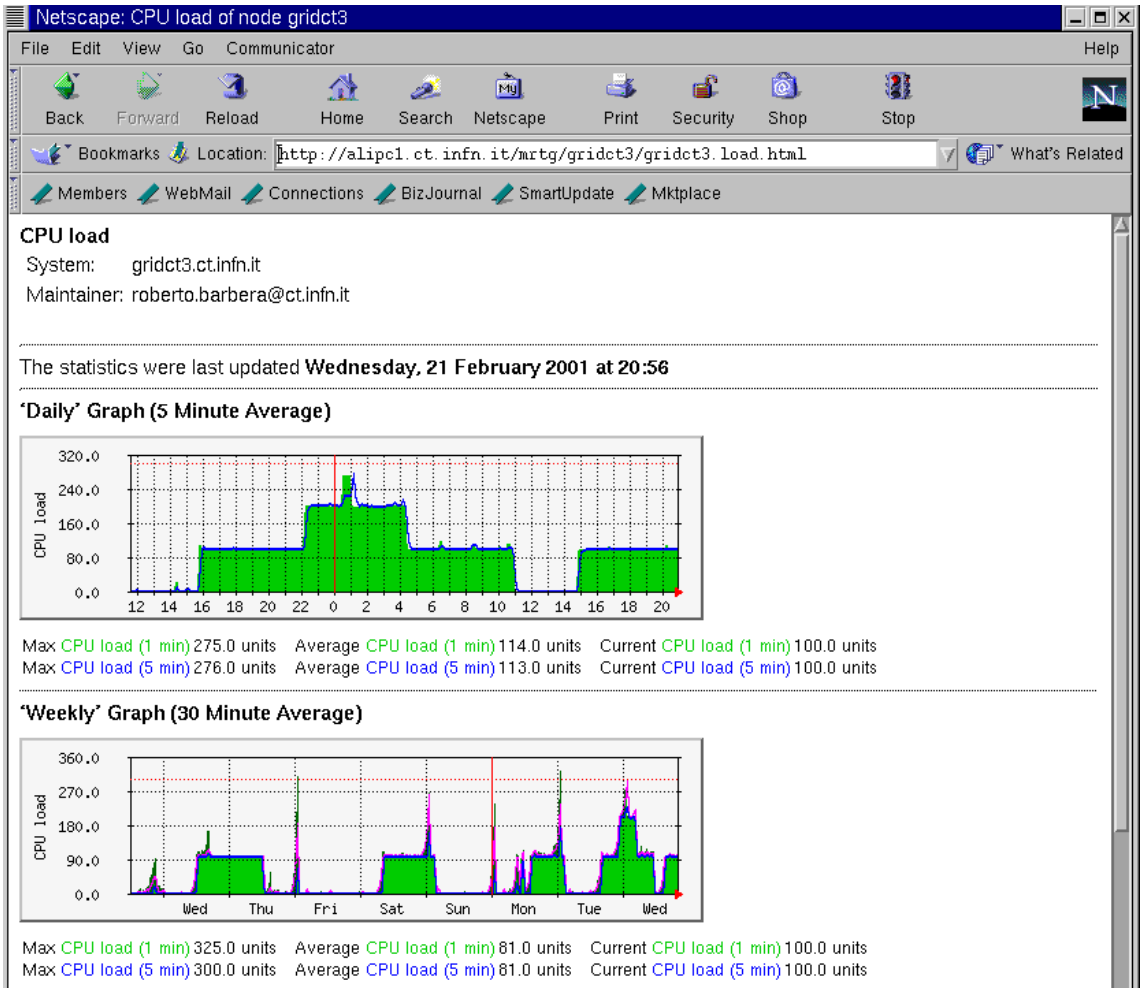

Figure 25: The CPU load monitoring of a node of the ALICE test-bed.

References

[1] See `http://www.cern.ch/ALICE`.

[2] All information concerning the ALICE Off-line Project can be found at the URL: `http://alisoft.cern.ch/offline`.

[3] All information about the DataGRID project can be found at the DataGRID Project home page: `http://www.datagrid.cnr.it/`.

[4] An invaluable starting point to get information on computational "GRIDs" is the book "The GRID: blueprint for a new computing infrastructure", edited by I. Foster and C. Kesselman. Other information can also be found at the URL: `http://www.egrid.org`.

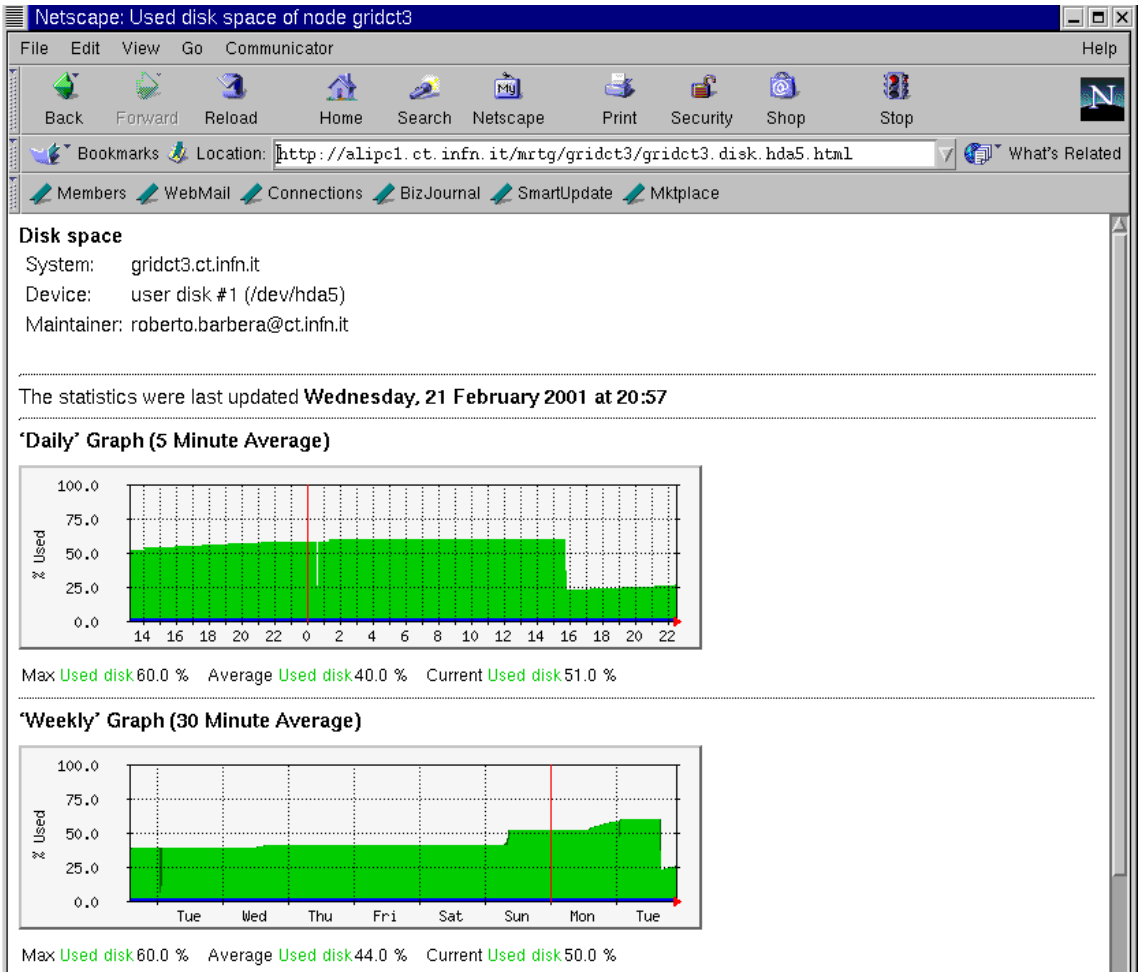

Figure 26: The disk space monitoring of a node of the ALICE test-bed.

[5] All information about `Globus` can be found at the `Globus` Project home page: `http://www.globus.org`. Information about the `Globus` evaluation inside the INFN can be found at the URL: `http://www.infn.it/globus/`.

[6] All information about the INFN GRID project can be found at the INFN GRID project home page: `http://www.infn.it/grid/`.

[7] Information about `EnginFrame` (the help pages are continuously improved) can be found at the URL:
`http://www.enginframe.com/sentinel/schema.xml`.

[8] All information about LSF can be found at the Platform home page: `http://www.platform.com`.

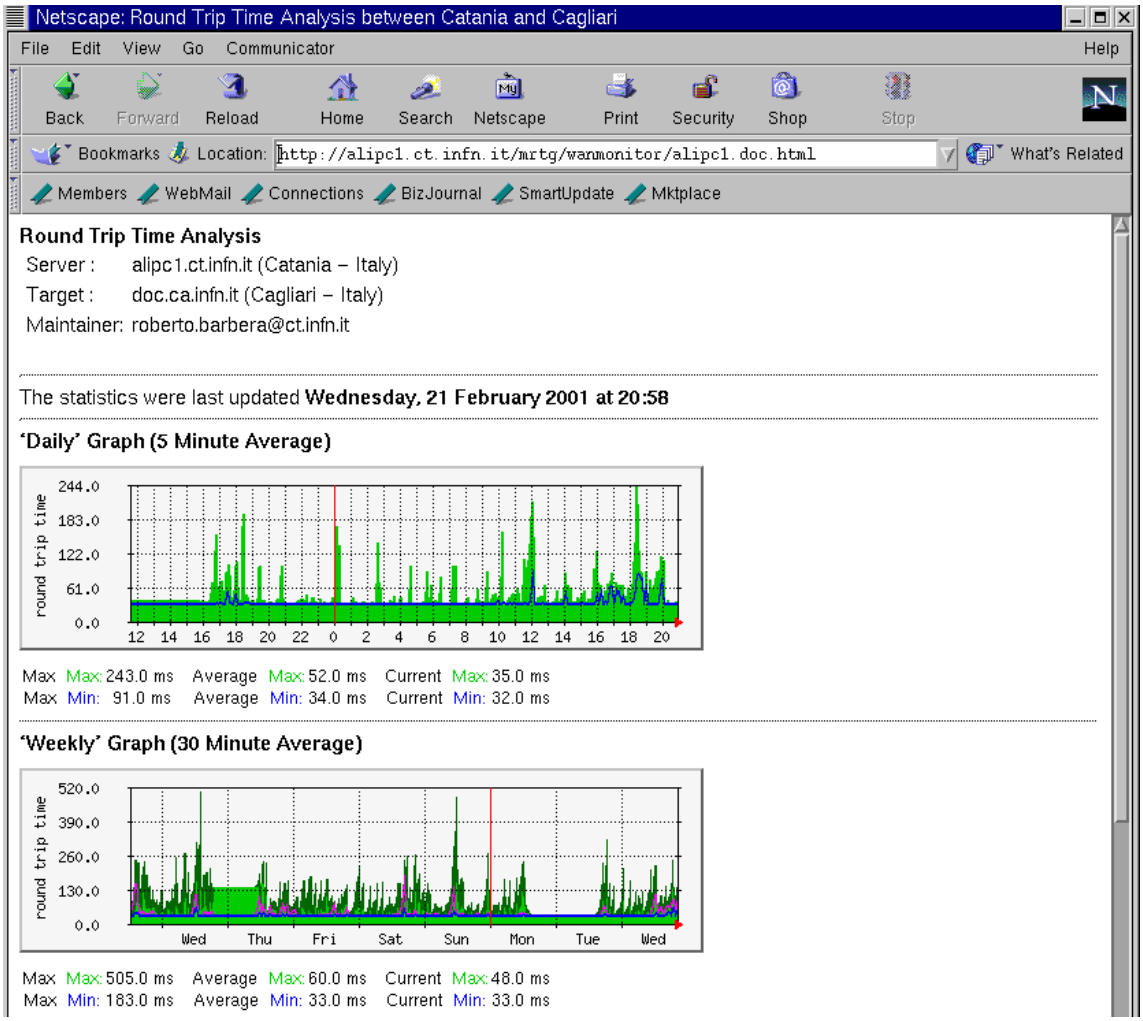

Figure 27: The network latency monitoring between two nodes of the ALICE testbed.

[9] All information concerning the `Globus` commands and services described in this note and, in particular, the `Globus` Resource Specification Language can be found in the `Globus` Quick Start Guide at the URL: `http://www.globus.org/toolkit/documentation/QuickStart.pdf`.

[10] All information about JAVA can be found at the page: `http://java.sun.com`.

[11] Documentation about the requirements can be found at the `XML` Apache Project at the URL: `http://xml.apache.org`. This page also contains the links to the required packages' home pages while the `JAVA` Development

Kit can be downloaded from the URL:
http:// http://java.sun.com/products/jdk/1.2/ for the .

[12] All information concerning the current version of LDAP Browser/Editor can be found at the URL:
http://www-unix.mcs.anl.gov/~gawor/ldap/.

[13] The INFN LDAP server can be accessed at the URL:
http://bond.cnaf.infn.it:389.

[14] See http://www.mi.infn.it/~lobiondo/gridmap/.

[15] All information about the INFN GRID Distribution Toolkit can be found at the URL: http://www.pi.infn.it/GRID/dist/.

[16] The INFN MDS server can be accessed at the URL:
http://bond.cnaf.infn.it/cgi-bin/mdsbrowse1.pl.

[17] All information about MRTG can be found at the MRTG home page:
http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/.

[18] All information about AliRoot can be found at the ALICE Off-line Project home page at the URL: http://alisoft.cern.ch/offline/.