OpenDLibG: Extending OpenDLib by Exploiting a gLite Grid Infrastructure

Leonardo Candela, Donatella Castelli, Pasquale Pagano, and Manuele Simi

Istituto di Scienza e Tecnologie dell'Informazione "Alessandro Faedo" – CNR
Via G. Moruzzi, 1 - 56124 PISA - Italy
{candela, castelli, pagano, simi}@isti.cnr.it

Abstract. This paper illustrates how an existing digital library system, OpenDLib, has been extended in order to make it able to exploit the storage and processing capability offered by a gLite Grid infrastructure. Thanks to this extension OpenDLib is now able to handle a much wider class of documents than in its original version and, consequently, it can serve a larger class of application domains. In particular, OpenDLib can manage documents that require huge storage capabilities, like particular types of images, videos, and 3D objects, and also create them on-demand as the result of a computational intensive elaboration on a dynamic set of data, although performed with a cheap investment in terms of computing resource.

1 Introduction

In our experience in working with digital libraries (DLs) we have often had to face the problem of resources scalability. Recent technology progresses make it now possible to support DLs where multimedia and multi-type content can be described, searched and retrieved with advanced services that make use of complex automatic tools for feature extraction, classification, summarization, etc. Despite the feasibility of such DLs, the actual use of them is still limited because of the high cost of the computer resources they require. Thus, for example, institutions that need to automatically classify images or 3D objects are forced to afford the cost of large processing capabilities even if this elaboration is only sporadically done. In order to overcome this problem, a couple of years ago we decided to start investigating the use of Grid technologies for supporting an effective handling of these objects. By using the features of the Grid several institutions can share a number of storage and computing resources and use them on-demand, on occasion of their temporary need. This organization allows minimizing the total number of resources required and maximizing their utilization.

Our attempt of exploiting Grid technologies is not isolated. Others are moving in the same direction even if with different objectives. Widely used content repository systems, like DSpace [18] and Fedora [13] as well as DLs, like the Library of Congress, are presently using the SDSC Storage Resource Broker [17] (SRB) as a platform for ensuring preservation and, more in generally, the long term availability of the access to digital information [15, 16].

Chershire3 [14], is an Information Retrieval system that operates both in single processor and in Grid distributed computing environments. A new release of this system capable of processing and indexing also documents stored in the SRB via their inclusion in workflows has been recently designed.

DILIGENT [6] aims at generalizing the notion of sharing proposed by the Grid technologies by creating an infrastructure that connects not only the computing and storage resources but, more generally, all the resources that compose a DL, i.e. archives of information, thesauri, ontologies, tools, etc. By exploiting the functionality provided by DILIGENT, digital libraries will be created on-demand by exploiting the resources connected through this particular infrastructure.

This paper describes how we have extended an existing DL system, OpenDLib [4], in order to make it able to exploit the sharing of storage and processing capabilities offered by a gLite Grid infrastructure[12] for effectively handling new document types. The system resulting from this extension, named *OpenDLibG*, can manage documents that require huge storage capabilities, like particular types of images, videos, and 3D objects, and also create them on-demand as the result of a computational intensive elaboration on a dynamic set of data. The novelty of this system with respect to its predecessor is that, by exploiting the on-demand usage of resources provided by the Grid, it can provide reliable, scalable and high throughput functionality on complex information objects without necessarily large investments on computing resources.

The paper presents the technical solution adopted by highlighting not only the potentialities related to the use of a Grid infrastructure in the particular DL application framework, but also the aspects that have to be carefully considered when designing services that exploit it. The additional features offered by this new version of the OpenDLib system are illustrated by presenting a real application case that has been implemented to concretely evaluate the proposed solution.

The rest of the paper is organized as follows: Section 2 briefly introduces OpenDLib and gLite; Section 3 provides details on the technical solution that has been implemented; Section 4 describes the new functionality OpenDLibG is able to offer and illustrates this functionality by presenting an implemented usage scenario; and finally, Section 5 contains final remarks and plans for further extensions.

2 The Framework

In this section we present a very brief overview of the OpenDLib and gLite technologies by focussing on those aspects that are relevant for describing OpenDLibG.

2.1 OpenDLib

OpenDLib [4] is a digital library system developed at ISTI-CNR. Its is based on a Service-Oriented Architecture that enables the construction of networked DLs hosted by multiple servers belonging to different institutions. Services implementing the DL functionality communicate through a HTTP-based protocol named OLP [5]. These services can be distributed or replicated on more than one server and can be logically organised as in Figure 1.

The *Collective Layer* contains services performing the co-ordination functions (e.g. mutual reconfiguration, distribution and replication handling, work-load distribution) on the services federation. In particular, the *Manager* Service maintains a continually updated status of the DL service instances and disseminates it on request to all the other services that use this information to dispatch message requests to the appropriate service instances. Thanks to this functionality, each service instance does not need to know where the other instances are located and how to discover the appropriate instances to call.

The *DL Components* includes services implementing DL functions. The basic OpenDLib release offers services to support the description, indexing, browsing, retrieval, access, preservation, storage, and virtual organization of documents. In particular, the storage and the dissemination of documents is handled by the *Repository* service.

The *Workflows* provides functionality implemented through workflows, i.e. structured plans of service calls. In particular, this area includes the *Library Manager* which manages and controls the submission, withdrawal and replacement of documents.

The *Presentation* contains services implementing the user front-ends to the other services. It contains a highly customisable *User Interface* and an *OAI-PMH Publisher*.

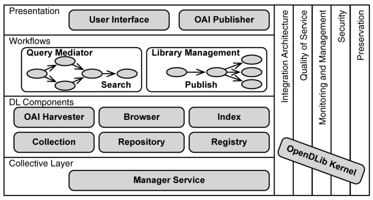


Fig. 1. The OpenDLib Layered Architecture

The *OpenDLib Kernel* supports all the above services by providing mechanisms to ensure the desired quality of service.

These services can be configured by specifying a number of parameters, like metadata and document formats, user profile format, query language, etc. The set of services illustrated above can be extended by including other services that implement additional application-specific functionality.

The OpenDLib services interact by sharing a common information objects model, *DoMDL* [2]. This model can represent a wide variety of information object types with different formats, media, languages and structures. Moreover, it can represent new types of documents that have no physical counterpart, such as composite documents consisting of the slides, video and audio recordings of a lecture, a seminar or a course. It can also maintain multiple editions, versions, and manifestations of the same document, each described by one or more metadata records in different formats. Every manifestation of the digital object can be either locally stored, or retrieved from a remote server and displayed whether at run time or in its remote location. A manifestation can also be a reference to another object manifestation; through this mechanism, data duplication can be avoided.

2.2 gLite

gLite [12] is a Grid middleware recently released by EGEE [7], the largest Grid infrastructure project currently being funded in Europe.

The role of gLite is to hide the heterogeneous nature of both the *computing elements* (CEs), i.e. services representing a computing resource, and *storage elements* (SEs), i.e.

services representing a storage resource, by providing an environment that facilitates and controls their sharing.

The services constituting the gLite software are logically organized as in Figure 2.

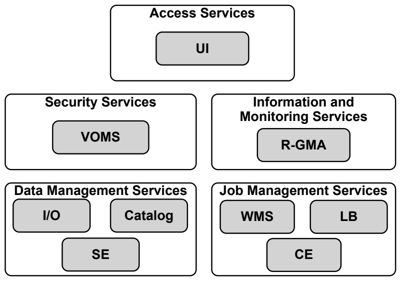


Fig. 2. The gLite Services

The *Job Management Services* is in charge of managing *jobs* and *DAGs*[1]. The core components of this subsystem are the *Computing Element*, the *Workload Manager* (WMS), and the *Logging and Bookkeeping* services. The former represents a computing resource and provides an interface for job submission and control. It is worth noting that the back end of the CE is composed by a set of computational resources managed by a Local Resource Management System (LRMS), e.g. Torque, Condor[2]. The Workload Manager is the subsystem whose main role is to accept requests of job submission and forward them to the appropriate CEs. The Logging and Bookkeeping service is in charge of tracking jobs in terms of *events* - e.g. submitted, running, done - gathered from various WMSs and CEs.

The *Data Management Services* is in charge of managing data and file access. gLite assumes that the granularity of data is on file level and that the access is controlled by Access Control Lists. The main services are the *gLite I/O*, the *Storage Element*, and the *Data Catalog*. The former provides a POSIX-like file I/O API, while the Storage Element represents the back end storage resource and can be implemented with various Storage Resource Managers, e.g. dCache[3], DPM[4]. The Data Catalogue allows to perceive the storage capacity of the infrastructure as a single file system.

The *Security Services* is in charge of dealing with authentication, authorization and auditing issues. Actually, the Virtual Organization Membership Service (VOMS) is the main service dealing with these issues. Other aspects are regulated via well known standards and technologies, e.g. X.509 Proxy Certificates [19], Grid Map Files.

The *Information and Monitoring Services* discovers and monitors the resources forming the infrastructure. The main service is the Relational Grid Monitoring Architecture (R-GMA), a registry supporting the adjunction and the removal of data about the resources constituting the infrastructure.

The *Access Services* enables end-users to have access to and use the resources of the infrastructure. Its main component is the User Interface (UI), a suite of clients and APIs enabling users to perform the common user tasks of a gLite based infrastructure, e.g. store and retrieving files, run jobs and monitor on their status.

[1] In gLite terminology jobs are an application that can run on a CE, and DAGs are directed acyclic graphs of dependent jobs.

[2] `http://www.cs.wisc.edu/condor/`

[3] dCache is accessible at `http://www.dcache.org`

[4] DPM information can be found at `http://wiki.gridpp.ac.uk/wiki/Disk_Pool_Manager`

3 OpenDLibG: gLite Integration and Exploitation

The OpenDLib document model is flexible enough to represent a large variety of complex information objects that, if largely employed, could change the way in which research is done. By exploting the functionality built on this model multimedia objects can be composed with table, graphs or images generated by elaborating a large amount of raw data, videos can be mixed with text and geographic information, and so on. Even if the support to this type of complex information objects is theoretically possible with OpenDLib, in practice it turns out to be unrealistic due to the large amount of computing and storage resources that have to be employed to provide performance acceptable by users. Our decision to extend OpenDLib by making it able to exploit the storage and processing capabilities provided by a gLite-compliant infrastructure was mainly motivated by the aim of overcoming this heavy limitation. In the rest of this section we describe the components that we have added, how they have been integrated in the architecture, and the difficulties that we faced in performing this integration.

3.1 The Integrated Architecture

In order to equip OpenDLib with the capabilities required to exploit a gLite-compliant infrastructure we designed the following new services:

- *gLite SE broker*: interfaces OpenDLib services with the pool of SEs made available via the gLite software and optimises their usage.
- *gLite WMS wrapper*: provides OpenDLib services with an interface to the pool of gLite CEs and implements the logic needed to optimize their usage.
- *gLite Identity Provider*: maps the OpenDLib user and service identities onto gLite user identities that are recognized and authorized to use gLite resources.
- *OpenDLib Repository++*: implements an enhanced version of the OpenDLib Repository service. It is equipped with the logic required to manage and optimize the usage of both OpenDLib repositories and gLite SEs as well as to manage novel mechanisms for the dynamic generation of document manifestations.

The architecture of the resulting system is depicted in Figure 3.

Thanks to the extensibility of the OpenDLib application framework the integration of these services has been obtained by only modifying the configuration of some of the existing services without any change in their code. In particular, the OpenDLib Manager Service has been appropriately configured to provide information about the new services and to disseminate new routing rules. These rules enable the OpenDLib UI to interact with instances of the OpenDLib Repository++ service in a completely transparent way for both the submission of, and the access to, documents while the Repository service is only accessed through its new enhanced version.

We explicitly chose to build the enhanced version of the Repository service as an abstraction of the basic version. It does not replace the original service because not all digital libraries require a Grid-based infrastructure. However, this new service maintains all the main characteristics of the basic version and, in particular, it can be replicated and/or distributed in the infrastructure designed to provide the DL application. Finally, the Repository++ can manage a multitude of basic Repository services while a same basic Repository service can accept requests coming from different Repository++.

In the rest of this section we present each of the new services in detail.

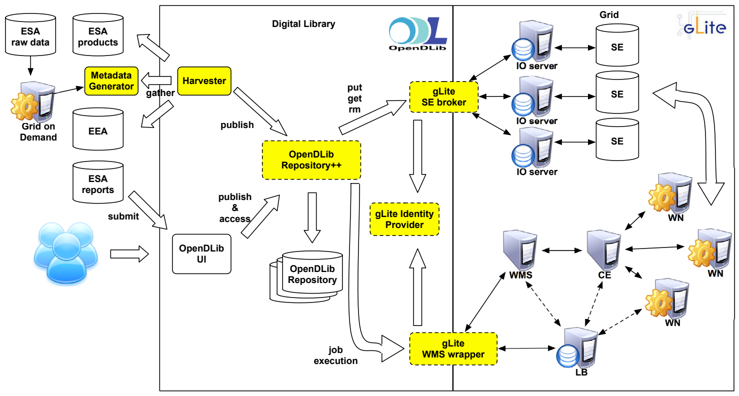


Fig. 3. OpenDLib integrate with gLite: the Architecture

The gLite SE broker. It provides the other OpenDLib services with the capability of using gLite based storage resources. In particular, this service interfaces the *gLite I/O server* to perform the storage and withdrawal of files and the access to them.

In designing this service one of our main concerns was to overcome two problems we have discovered experiencing with the current gLite release: (*i*) the inconsistency between catalog and storage resource management systems and (*ii*) failure in the access or remove operations without notification.

Although the gLite SE broker could not improve the reliability of the requested operations we designed it to: (*i*) monitor its requests, (*ii*) verify the status of the resources after the processing of the operations, (*iii*) repeat file registration in the catalog and/or storage until it is considered correct or unrecoverable, (*iv*) return a valid message reporting the exit status of the operation. The feasibility of this approach was validated by the small resulting delay experimentally measured as well as by real users judgements.

In order to appropriately exploit the great number of resources provided by the infrastructure, the gLite SE broker service was designed to interface more than one I/O server for distributing storage and access operations. In particular, this service can be configured to support three types of storage strategies for distributing files among the I/O servers, namely: (*i*) round-robin, (*ii*) file-type-based, which places the files of a certain type on a predefined set of I/O servers, and (*iii*) priority-based, which is useful to enhance one of the previous strategies with an identified prioritized list of I/O servers ordering the requests to them. It is worth noting that the service can also dynamically rearrange the prioritized list by taking into account performance characteristics, e.g. the time and the number of failures in executing I/O actions.

Inspired by the RAID technology[5] we designed the gLite SE broker to support the *RAID 1* modality that mirrors each file by creating a copy of it on two or more servers. This feature is activated by default but it can be explicitly turned off at configuration time.

The *RAID 0* modality, a.k.a. striped, that splits files on two or more servers and the possibility to select the appropriate modality for each file at the submission time is under investigation.

[5] A Redundant Array of Independent Disks, a.k.a. Redundant Array of Inexpensive Disks.

The gLite WMS wrapper. It provides the other OpenDLib services with the computing power supplied by gLite CEs. In particular, this service offers an interface for managing *jobs* and *DAGs* with an abstraction level higher than that provided by gLite.

The gLite WMS broker has been designed to: *(i)* deal with more than one WMS, (*ii*) monitor the quality of service provided by these WMSs by analyzing the number of managed jobs and the average time of their execution, and, finally, (*iii*) monitor the status of each submitted job querying the *Logging and Bookkeeping* service. As a consequence of the implemented functionality, the gLite WMS service represents a single point of access to the computing capabilities provided by the WMS services and to the monitoring capabilities provided by the LB services. This approach decouples the gLite infrastructure from the OpenDLib federation of services while hiding their characteristics. Moreover, by exploiting the features provided by the OpenDLib application framework, the gLite WMS broker service can be replicated in a number of different service instances, each managing the same set of gLite services, or can be distributed over a number of different service instances, each managing a different pool of gLite services.

In implementing this component we provided both a round-robin and a priority based scheduling strategies to manage the distribution of jobs to WMSs. In particular, the second approach represents an enhancing of the first one because it identifies a priority list of WMSs ordering the requests to them. It is still under investigation the possibility of automatically manipulating this priority in order to take into account performance metrics such as the time and the number of failures.

Finally, we equipped the service with a basic fault tolerance capability in performing job submission tasks that repeats the execution in case of failure.

gLite Identity Provider. The mechanisms that support authentication and authorization in OpenDLib and gLite are different. The two systems have been designed with the aim to satisfy different goals in a completely different usage scenarios: OpenDLib operates in a distributed framework where the participating institutions collaborate and share the same rules and goals under the supervision of a distributed management, while gLite has to work in an environment where policies and access rules are managed differently by the participating institutions. OpenDLib builds its own authentication mechanism on user name and password, while gLite builds it on X.509 Certificates. Moreover, the authorization mechanisms for assigning policies to users are proprietary in OpenDLib while they are based on the Virtual Organization mechanism and Grid Map Files in a gLite based infrastructure. In order to reconcile these authentication and authorization frameworks a service able to map OpenDLib identities on gLite identities was introduced. The main characteristics of this service are:

- it generates the Proxy Certificates [19] that are needed to interact with gLite resources. In order to support this functionality it has to be equipped with the appropriate pool of personal certificates that, obviously, must be stored on a secure device.
- it can be configured to establish the mapping rules for turning OpenDLib identities into gLite identities.

The OpenDLib Repository++. This service was designed to act as a *virtual repository*, capable of the same operations as those required to store and access documents in a traditional OpenDLib DL. In this way the other services of the infrastructure do not need either to be aware of this service's enhanced capabilities nor to be redesigned and re-implemented. Despite the public interface of this service completely resembles the *Repository* interface, its logic is completely different because it does not store any content locally, instead, it relies on the storage facilities provided by both the OpenDLib Repository and the gLite infrastructure via the gLite SE broker.

In designing this component we decided to make the strategy for distributing content on the two kinds of storage systems configurable.

The configuration aspects exploit the DoMDL management functionality allowing any supported manifestation type to be associated with a predefined workflow customising storage, access, and retrieve capabilities. It is thus possible to design and implement the most appropriate processes for each new type of raw data managed by the DL and easily associate it with the manifestation type. In the current version, one workflow to store, access, and retrieve files through the described gLite wrappers has been implemented. For instance, it is possible to configure the Repository++ service in order to maintain all metadata manifestations on a specific OpenDLib Repository instance, a certain manifestation type on another OpenDLib Repository, while raw data and satellite products that are accessed less frequently and require a huge amount of storage can be stored on a SE provided by the gLite based infrastructure. The characteristics of the content to be stored should drive the designer in making the configuration. Usually, manifestations that require to be frequently accessed, or that need to be maintained under the physical control of a specific library institution, should be stored on standard OpenDLib Repository services. On the contrary, content returned by processes, that either is not directly usable by the end-user, or can be freely distributed on third-party storage devices should be stored on gLite SEs.

Cryptography capabilities are under investigation to mitigate the problems mostly related to the copyright management for storing content on third-party devices.

Another important feature added to the enhanced repository is the capability of associating a job or a DAG of jobs with a manifestation. This feature makes it possible to manage new types of document manifestations, i.e., manifestations dynamically generated by running a process at access time. The realisation of such extension has been quite simple in OpenDLib thanks to DoMDL. In fact, DoMDL is able to associate a URI of a specific task with a manifestation. In this case, this task uses the gLite WMS wrapper for executing a process customized with the information identifying the job/DAG to be run together with the appropriate parameters.

An example of the exploitation of this functionality is given in the following section.

4 OpenDLibG in Action: An Environmental DL

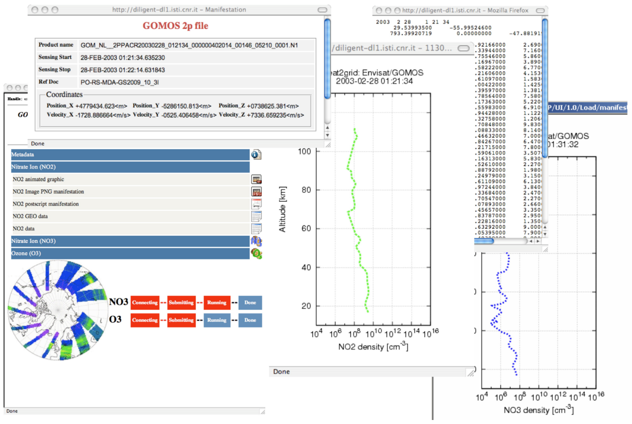
Stimulated by the long discussions we had with members of the European Space Agency (ESA)[6], we decided to experiment the construction of an OpenDLibG DL for supporting the work of the agencies that collaboratively work at the definition of environmental conventions. By exploiting their rich information sources, ranging from raw data sets to

[6] These discussions and the corresponding requirements where raised mainly in the framework of the activities related to the DILIGENT project.

maps and graphs archives, these agencies periodically prepare reports on the status of the environment. Currently, this task is performed by first selecting the relevant information from each of the multiple and heterogeneous sources available, then launching complex processing on large amount of data to obtain graphs, tables and other summary information and, finally, producing the required report by assembling all the different parts together. This process repeated periodically requires a lot of work due to the complexity of interfacing the different sources and tools. Despite the effort spent, the resulting reports do not always met the requirements of the their users since they present to its readers a picture of the environmental status at the time of the production of the report and not at time in which the information reported is accessed and used. To overcome this problem and, more generally, to simplify the generation of the environmental reports we created an OpenDLibG DL prototype.

From the architectural point of view, the OpenDLibG components of this DL are hosted on three servers. The first server is publicly accessible and hosts the User Interface service that allows end-users to easily interact with a human-friendly interface. The second and third servers are protected behind a firewall and host the basic and the extended OpenDLib services respectively.

As far as the Grid infrastructure is concerned, the OpenDLibG environmental DL exploits the DILIGENT gLite infrastructure. This infrastructure consists of five sites located in Pisa (Italy), Rome (Italy) Athens (Greece), Hall in Tyrol (Austria) and Darmstadt (Germany). Each site provides storage and computational capabilities for a total of 41 Processors, 38,72 GB RAM, and 3,28 TB disk space. For the scope of this DL, we decided to exploit only two storage elements based on dCache and other two storage elements based on DPM.


http://diligent-dl1.isti.cnr.it – Manifestation

GOMOS 2p file

Product name	GOM_NL__2PPACR20030228_012134_000000402014_00146_05210_0001.N1
Sensing Start	28-FEB-2003 01:21:34.635230
Sensing Stop	28-FEB-2003 01:22:14.631843
Ref Doc	PO-RS-MDA-GS2009_10_3I

Coordinates

Position_X +4779434.623<m> Position_Y -5286150.813<m> Position_Z +0738625.381<m>
Velocity_X -1728.886664<m/s> Velocity_Y -0525.406458<m/s> Velocity_Z +7336.659235<m/s>

Done

Metadata
Nitrate Ion (NO2)
NO2 animated graphic
NO2 Image PNG manifestation
NO2 postscript manifestation
NO2 GEO data
NO2 data
Nitrate Ion (NO3)
Ozone (O3)

NO3 Connecting -- Submitting -- Running -- Done
O3 Connecting -- Submitting -- Running -- Done

Fig. 4. A GOMOS Document

In this experimental environmental DL the Repository service has been configured to manage DoMDL instances that are able to maintain both information objects selected from third-party information sources - whose content is imported/linked in/to the DL - and information objects whose manifestations are generated on-demand using a registered workflow that invokes the gLite WMS Wrapper for executing specific elaborations.

This DL provides the data, the documents, the dynamically generated reports, and any other content and services deemed as relevant with respect to the day-by-day activity of people who have to take decisions on environmental strategies. In particular, the prototype can: (*i*) manage environmental reports, workshops proceedings and other types of documents relevant to the Earth Observation community collected from different and heterogenous information sources; (*ii*) deal with added value satellite products like chlorophyll concentration maps and mosaics; (*iii*) dynamically produce reports related to certain world regions and time periods; (*iv*) use the gLite job management facilities to produce Nitrate and Ozone profiles from satellite products, thus making experiments on the management of such data; (*v*) support search and use of a number of relevant external data, services, and resources concerned with Earth Science, like glossaries, gazetteers, and other digital libraries of interest.

In particular, the above information objects have been obtained by harvesting: (*i*) documents gathered or linked from external information sources like MFSTEP monthly reports[7] and the European Environment Agency[8] reports, briefings, indicators and news, (*ii*) high resolution satellite images both directly acquired and dynamically generated, and (*iii*) level two ENVISAT-GOMOS products[9] containing raw data on ozone, temperature, moisture, NO_2, NO_3, $OClO$, O_3 measures collected by the GOMOS sensor.

Figure 4 shows an example of the novel type of documents that can be managed by this DL. This document is composed by (*i*) a metadata view containing descriptive information like the start and stop sensing dates and the coordinates of the geographical area the document refers to and (*ii*) three products defined via appropriate workflows whose manifestations are generated by running workflows on the Grid infrastructure. These workflows exploit the BEAT2GRID application, provided by the ESA organisation and adapted by us to run on a gLite based infrastructure, and the appropriate operations for gathering from the Grid the raw data to be elaborated, storing on the Grid the obtained products, and linking them as document manifestations. Such workflows generate geolocation information extracted from the raw data; the NO_2/NO_3 image profile information showing the density with respect to the altitude; the NO_2/NO_3 profile information comprising date, time, longitude and latitude of tangent point, longitude and latitude of satellite; the ozone density with respect to the altitude and the ozone density covariance. Each of such products represent a manifestation of a product view. According to the document definition, each product manifestation can be retrieved from the Grid or dynamically generated at access time. To give access to these complex objects a specialised user interface has been designed. It is capable capable to start the product generation process, progressively show the status of the workflow execution and, once products are generated, give access to them.

[7] `http://www.bo.ingv.it/mfstep/`

[8] `http://www.eea.eu.int/`

[9] ENVISAT (ENVIronment SATellite) is an ESA Earth Observation satellite whose purpose is to collect earth observations: it is fitted with 10 sensors ASAR, MERIS, AATSR, RA-2, MWR, DORIS, GOMOS, MIPAS, SCIAMACHY, LRR and other units. Detailed information about the ENVISAT satellite can be found at `http://envisat.esa.int/`

It is worth noting that the BEAT2GRID application is executed in a couple of minutes in a quite normal bi-processor entry-level server. However, standard DL based applications can not provide such functionality to end-users since hundreds of concurrent requests prove the limited scalability of a static infrastructure. OpenDLibG powered by the described gLite based infrastructure, instead, proves to manage tenths of concurrent requests with the same throughput as the single execution on a dedicated server, and to correctly manage a higher number of requests by using queue mechanisms. The same observation holds with respect to the storage capacity. Raw data and intermediate processing results require an huge amount of storage space. Thanks to the Grid technology this space can be obtained on demand by relying on third party institutions while in the case of a standard DL it is needed to equip the DL with such amount of resources even if they are needed only for a limited time period.

This experimentation can be considered the first step in the exploitation of Grid enabled DLs. Moreover it represents a great opportunity for both users and digital library developers to share views and language, to express needs via practical examples, to understand capabilities for future exploitation, to access practical progress, to evaluate opportunities and alternative solutions, to support technical decisions and, last but not least, to develop critical interfaces.

5 Conclusions and Lesson Learned

This paper has described OpenDLibG, a system obtained by extending the OpenDLib digital library system with services exploiting a gLite Grid infrastructure. As a result of this extension OpenDLibG can provide both a more advanced functionality on novel information objects and a better quality of service without requiring a very expensive infrastructure.

We strongly believe that the new information objects described in this paper will play an increasingly important role in the future as they can contribute to revolutionize the way in which many communities perform their activities. In this paper we have shown only an example of the exploitation of such documents, but many others have been suggested us by the many user communities we are in contact with.

The integration of OpenDLib with a Grid infrastructure not only makes it possible to handle the new type of objects but it also supports any functionality whose implementation requires intensive batch computations. For example, periodic complex feature extraction on large document collections or generation and storage of multiple and alternative manifestations for preservation purposes can similarly be supported while maintaining a good quality of service. Our next future plan is to extend the system with novel and distributed algorithms for providing DL functionality relying on the huge amount of computing and storage power provided by the Grid.

While carrying out this experience we have learnt that there are a number of aspects that have to be carefully considered in designing a DL system that exploits a Grid infrastructure. In this framework resources are provided by third-parties and there is a lack of any central control on their availability. These resources can disappear or become unavailable without informing any central authority that, therefore, has no means to prevent it. This problem is made worst by the lack of advanced reservation, i.e. the possibility for a resource user to agree with the resource provider on the availability of a resource for a well established time period and on a given quality of service. This feature is a long term goal in the Grid research area and it is expected that it will be provided in future releases of Grid middleware. This lack has strong implications on the reliability of the Grid resources usage. For example, a document stored on a single SE can be lost if the SE is removed from the Grid by its provider. Appropriate measures have to be taken to reduce the risk induced by this lack. For example, a DL service must be designed in such a way that if the CE running one of its processes disappears, it must be able to recover this malfunction. Other aspects to be carefully taken into account are related to performance. Some of them apply to any Grid infrastructure, while others are more specific and relate to the gLite software and its current release. Perhaps the most important among these aspects is concerned with the communication overhead that arises when using resources spread over the Net. In this context, where resources are SEs and CEs, the decision to ask third-party for the storage or the processing capabilities must be carefully evaluated, i.e. the enhancement obtained must be compared with the overhead needed and the right trade-off among these aspects must be discovered.

Acknowledgments. This work is partially funded by the European Commission in the context of the DILIGENT project, under the 2nd call of FP6 IST priority.

We thank all the DILIGENT partners that have contributed with their suggestions and expertise to this work. Particular thanks go to ESA, which provided us the requirements for live documents, the real data and the specific applications for our experimentation; CERN which supported us in our understanding of the gLite technology; and FhG-IPSI which studied the problem of dynamically defining the structure and visualization of the on-demand documents and which collaborated with CNR for the setting up of the experimentation.

References

1. W. Y. Arms. *Digital Libraries*. The MIT Press, September 2001.
2. L. Candela, D. Castelli, P. Pagano, and M. Simi. From Heterogeneous Information Spaces to Virtual Documents. In *Proceedings of the 8th International Conference on Asian Digital Libraries, ICADL 2005, Bangkok, Thailand, December 2005*, pages 11–22. Springer, 2005.
3. L. Candela, D. Castelli, P. Pagano, and M. Simi. Moving Digital Library Service Systems to the Grid. In *Peer-to-Peer, Grid, and Service-Orientation in Digital Library Architectures*, number 3664 in Lecture Notes in Computer Science, pages 236 – 259. Springer Verlag, 2005.
4. D. Castelli and P. Pagano. A System for Building Expandable Digital Libraries. In *ACM/IEEE Joint Conference on Digital Libraries (JCDL 2003)*, pages 335–345. Springer-Verlag, 2003.
5. D. Castelli and P. Pagano. The OpenDLib Protocol. Technical report, Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR, 2004.
6. DILIGENT. A DIgital Library Infrastructure on Grid ENabled Technology. `http://www.diligentproject.org`.
7. EGEE. Enabling Grids for E-science in Europe. `http://public.eu-egee.org`.
8. I. Foster. What is the Grid? A Three Point Checklist. *GRIDtoday*, 1(6), 2002.

9. I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan-Kaufmann, 2004.
10. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.
11. I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organization. *The International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.
12. gLite. Ligthweight Middleware for Grid Computing. `http://glite.web.cern.ch/`.
13. C. Lagoze, S. Payette, E. Shin, and C. Wilper. Fedora: An Architecture for Complex Objects and their Relationships. *Journal of Digital Libraries, Special Issue on Complex Objects*, 2005.
14. R. R. Larson and R. Sanderson. Grid-based digital libraries: Cheshire3 and distributed retrieval. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 112–113, New York, NY, USA, 2005. ACM Press.
15. R. W. Moore and R. Marciano. Building preservation environments. In M. Marlino, T. Sumner, and F. M. S. III, editors, *JCDL*, page 424. ACM, 2005.
16. A. Rajasekar, R. Moore, F. Berman, and B. Schottlaender. From Digital Preservation Lifecycle Management for Multi-media Collections. In *8th International Conference on Asian Digital Libraries, ICADL 2005, Bangkok, Thailand, December 2005*, pages 380–384. Springer, 2005.
17. A. Rajasekar, M. Wan, R. Moore, W. Schroeder, G. Kremenek, A. Jagatheesan, C. Cowart, B. Zhu, S.-Y. Chen, and R. Olschanowsky. Storage Resource Broker - Managing Distributed Data in a Grid. *Computer Society of India Journal, Special Issue on SAN*, 33(4):42–54, October 2003.
18. R. Tansley, M. Bass, and M. Smith. DSpace as an Open Archival Information System: Current Status and Future Directions. In *Proceedings of the 7th European Conference, ECDL 2003, Trondheim, Norway, August 2003*, pages 446–460. Springer-Verlag, 2003.
19. S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. RFC3820, IETF, The Internet Engineering Task Force, June 2004.