Performance Analysis of Information Services in a Grid Environment

Giovanni ALOISIO, Massimo CAFARO, Sandro FIORE, Italo EPICOCO, Maria MIRTO, Silvia MOCAVERO
Center for Adavanced Computing Technologies/ ISUFI University of Lecce & SPACI Consortium
Lecce, 73100, ITALY
{giovanni.aloisio, massimo.cafaro, sandro.fiore, italo.epicoco, maria.mirto, silvia.mocavero}@unile.it

ABSTRACT

The Information Service is a fundamental component in a grid environment. It has to meet a lot of requirements such as access to static and dynamic information related to grid resources, efficient and secure access to dynamic data, decentralized maintenance, fault tolerance etc., in order to achieve better performance, scalability, security and extensibility. Currently there are two different major approaches. One is based on a directory infrastructure and another one on a novel approach that exploits a relational DBMS. In this paper we present a performance comparison analysis between Grid Resource Information Service (GRIS) and Local Dynamic Grid Catalog relational information service (LDGC), providing also information about two projects (iGrid and Grid Relational Catalog) in the grid data management area .

Keywords: Information Service, Relational Data Model, LDAP, MDS, DGC.

1. INTRODUCTION

With the proliferation of the Internet comes the opportunity of aggregating and sharing a wide variety of heterogeneous and geographically distributed resources (supercomputer, storage systems, etc.) for solving large-scale computational problems in science and engineering. A grid environment [1] collects a lot of information (related to computational resources) in Information Services, providing a standard mechanism for publishing and discovering resource status and configuration information.
The Globus Toolkit [2] includes a set of information service components collectively referred to as the Monitoring and Discovery Service (MDS) [3]. It has a hierarchical structure that consists of three main components: GIIS (Grid Index Information Service), GRIS (Grid Resource Information Service) and IPs (Information Providers). MDS is based on a hierarchical data model and it allows managing static and dynamic information about the status of a computational grid and all its components.
Instead Dynamic Grid Catalog (DGC) [4] is an Information Service that leverages an emerging complementary approach based on the relational data model [5-7]. Its main components are: GDGC (Global Dynamic Grid Catalog relational information service), LDGC (Local Dynamic Grid Catalog relational information service) and IPs. LDGC is a relational information service related to a single grid resource and it allows storing information about a local machine. On the contrary GDGC, gathers information coming from one or more LDGCs.
In this paper we compare and analyze the performance of LDGC and GRIS. The paper is organized as follows: Section 2 describes the information service requirements, Section 3 the main differences between GRIS and LDGC, and Section 4 the experimental environment and results. In Section 5 we present a relational information service (iGrid) and in Section 6 we describe the Grid Relational Catalog project. We conclude the paper in Section 7.

2. INFORMATION SERVICE REQUIREMENTS

As pointed out by Global Grid Forum (GGF) [8-9] GIS working Group, and other previous works [10] a Grid Information Service should provide the following basic requirements:

- ✓ Performance: a centralized approach to store information related to all grid resources is not the best solution, indeed, it can lead to a performance bottleneck. Good performance associated to the enquiry protocol is also absolutely necessary due to the large number of queries that come to a GIS;
- ✓ Scalability: the number of clients in a Grid environment can grow exponentially, so a GIS infrastructure highly scalable is fundamental;
- ✓ Fault tolerance: we remember that component failure in grid environment is the rule;
- ✓ Stability: steadiness in query response time is another basic requirement;
- ✓ Extensibility: adding new kinds of information in the schema must be easy and fast. Users should be able to add new information items (creating their own information provider), and to publish them without radically changing the old system, but simply modifying the configuration GIS profile;
- ✓ Platform Independence: platform heterogeneity in grid environment is very common, so this additional complexity element can be solved designing a cross-platform Grid Information Service;
- ✓ Robustness and security: in order to address robust authentication (are you who you say you are), authorization (are you allowed to access the resources you are requesting for the tasks you want to perform) and communication protection requirements we adopt the GSI protocol [11]. It is an infrastructure that provides generic security services for applications that will be run on the grid. Security operations are also dependent on the particular operating system used. For example, SASL requires dynamically loaded libraries at runtime and this is not the rule for all operating systems;
- ✓ Dynamic Data: a good information service must be designed to meet a lot of requirements such as access to dynamic data (dynamic information related to grid resources) e.g. number of active processes, CPU usage etc.;
- ✓ Timestamp and Time To Live (TTL) attributes for information: to improve response time and maximize

flexibility, each provider's result may be cached for a configurable period. This cache time-to-live (TTL) is specified per-provider as part of the local GIS configuration. It is worth noting that timestamps and TTL estimates are not meaningful if a common timing mechanism is not used; thus some protocols like NTP must be mandatory;

- ✓ Efficient discovery, enquiry and delivery mechanisms: these mechanisms are necessary in a GIS. It is also important and fundamental that they perform well in a grid environment

3. GRIS AND LDGC INFORMATION SERVICE

In this paper we present a comparison between GRIS and LDGC Information Service.

In Globus Toolkit, MDS includes a standard, configurable information provider framework called a Grid Resource Information Service that supplies information about a specific resource and utilizes the LDAP [12] protocol. It can be customized by plugging in specific information sources.

A GRIS parses and dispatches each incoming request to one or more local information providers (we remember that GRIS will support the "lazy" behavior, that is, caches are only refilled on client query cache misses). Results are than merged back to the client, filtered by the GRIS to delete any objects that do not match the client's search space.

It is worth noting here that MDS 2 restricts clients to queries using the LDAP protocol, a fairly restrictive query language, and forces clients to download a lot of information and do subsequent client-side processing to find the desired data).

LDGC is a relational information service that has been designed to meet efficient and secure access to data in order to achieve better performance, security and scalability. It also allows to store historical information so that other components in a grid environment such as schedulers or grid resource broker, can make statistics or forecasts about the dynamic behavior of grid resources [4]. A LDGC can answer to queries coming from other systems on the grid asking for information about the local machine; this service is called White Page Service.

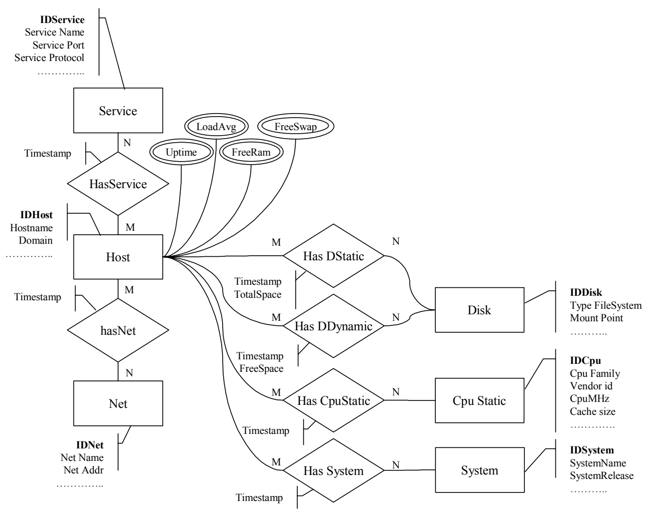

Figure 1. Entity relation diagram of information stored in LDGC information service.

An important feature of LDGC is the "cache pre-fetch". When TTL information causes data to be flushed from the cache, the LDGC will automatically restart the information providers in order to re-fill the cache. This stabilizes the performance of queries, since most of the time the cache will have the desired data.

The main differences between GRIS and LDGC information services are summarized in Table 1 as follows:

Table 1. Main differences between GRIS and LDGC Information Services

GRIS	**LDGC**
Hierarchical data model	Relational data model
Utilizes LDAP	Utilizes SQL
No aggregate selection (join)	Aggregate selection allowed
Explicit knowledge of the structure of the tree is required	Flat table
Directories may not support transactions	Transactions Supported
Directory service query language is a procedural language	SQL is a declarative language
Data are stored in an LDAP repository in native LDIF format. LDIF is also the format for data delivered to, and received from, MDS.	XML MultiQuery (proprietary format, see section 6) is the format for data delivered to LDGC.
To check for data consistency GRIS servers use GIS schema with default setting = NO	To check for MultiQuery data consistency DGC servers use a Document Type Definition (DTD) with default setting = YES
LDAP was designed to contain small records of information	LDGC relational database is designed to contain small and big records of information
GRIS performs poorly in the presence of frequent updates	Achievable high update rates and freshness
No complex queries	LDGC supports both simple and complex queries
GRIS do not provide support for data streams	Streaming supported
Historical information not stored	Historical information also stored
Not very efficient and scalable support for a large and growing number of data objects	Efficient and scalable support for a large and growing number of data objects
Need to have some client-side processing	Rich queries that allow to find data without having to do any client-side processing
Providers are restarted when a new query is received and information cache is expired	Providers are automatically restarted when information cache is expired

4. EXPERIMENTAL ENVIRONMENT AND RESULTS TEST

The test queries used in our experiments were related to many kinds of information. For instance in our tests we retrieved the following ones:

- ✓ Static Host Information (operating system, number of processors, etc.);

✓ Dynamic Host Information (load average, free memory Ram, free memory Swap, etc);
✓ Storage System Information (total disk space, available disk space, etc);
✓ Network Interface Information (machine names and addresses).

To make a performance comparison between Globus MDS GRIS 2.2 and LDGC 1.0 we did the following tests.

First we created in the Initialization Phase a Relational Database for the LDGC Information Service with the same information that we have in the GRIS schema.

A simple partial Entity-Relation (ER) [13] model can be seen in Figure 1.

Then we ran on the same machine GRIS and LDGC Information Providers (only core providers were considered) in order to generate information needed in the Population Phase. Finally, two applications (clients) were developed in order to contact and to query GRIS and LDGC servers in the Querying Phase. In this last phase, due to the not very large amount of data we note no considerable difference between performance related to different queries (both in GRIS using LDAP and in LDGC using SQL language).

In LDGC, the unique SQL operation time-wasting is a join, but joins between tables involve few tuples, so even changing queries, results were the same. Also in GRIS changing the entry point in the tree no considerable differences were found. We tested GRIS and LDGC with same reference-query related to the following processor information: CpuSpeedMhz, CpuVendor, CpuModel, CpuVersion, CpuFree1MinX100.

The first client application uses Globus API and LDAP library to query the GRIS server. Moreover, it uses Simple Authentication and Security Level (SASL - binding through GSI-GSS API) to establish a secure connection with the GRIS server.

To test GRIS using SASL we configured properly our GRIS server installing a required Globus certificate needed for mutual authentication. The second one uses Globus API and DGC library (version 1.0) to bind and query the DGC server. The security mechanism used in this application is based on the Grid Security Infrastructure (GSI) which enables the use of X509 certificates in order to provide authentication and authorization services.

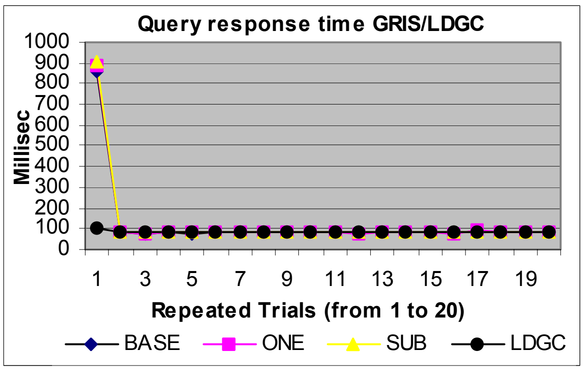

Figure 2. Query response time for GRIS (base, one, sub scopes) and LDGC

We choose not to use the *ldapsearch* and *dgcsearch* (tools that are respectively available in the Globus and DGC Information Service distributions) because using them would have resulted in timings affected by an unnecessary overhead (initial setup, parameter options parsing, etc.) and also because real grid-enabled applications will exploit directly the Globus API and LDAP and DGC libraries.

The experiment were performed on a PC (*sara.unile.it*) with a AMD 1.0GHz processor, 512MB of main memory, Linux Operating System (kernel 2.4) and a 30GB EIDE HD 7200RPM. We used a free RDBMS (PostgreSQL 7.4 in our tests, but this is not a fixed constraint in our LDGC implementation because we can use whatever RDBMS as for instance Oracle, MySQL, etc.) with default system settings.

There were no other considerable running applications during the experiments. To take into account the caching mechanisms, we proceeded issuing a first set of 20 queries to our GRIS server using the base scope. Then we waited for 20 minutes in order for the GRIS cache to expire (by default the expiry period for almost all the information providers is set to 1 or 15 minutes; there are three information providers for which the cache expire after 12 hours, so that the information provided was considered as always available).

Once again a new set of 20 queries was issued, this time with one as scope. After 20 minutes a set of 20 queries was issued using the sub scope. Finally a set of 20 queries (SQL) was issued to the DGC server.

The experiment was repeated several times during the day (peak time for computing resource with high workload) and late during the night (off-peak time, low workload) and we obtained the same results. For every client applications results were averaged in order to address accuracy test requirement.

We tested only the *Search* operation because it is the most frequent one related to the analyzed Information Services (other operations as Insert, Update and Delete are not considered in this comparison).

Referring to Figure 2 it is worth noting that caching mechanism used by GRIS is evident. As expected, for the GRIS, the first query in each set (scope *base one* and *sub*) takes significantly more time due to the empty cache; also expected is the increase in response time when querying the GRIS with scope going from *base* to *one* and finally to *sub*.

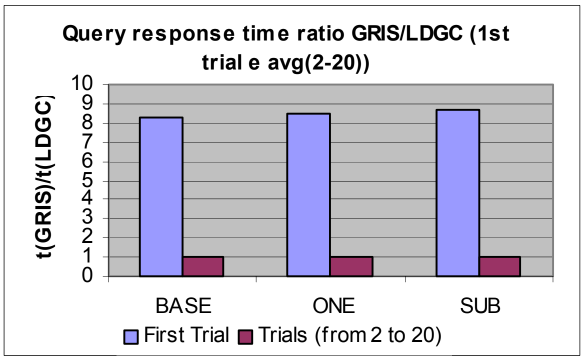

Figure 3. Query response time ratio (GRIS/LDGC) (repeated trials from 1 to 20)

On the contrary LDGC is more stable and the difference between the first query and the others is less evident. Referring to Figure 3, if we consider the query time ratio between GRIS and LDGC we obtain a very interesting information; indeed response time for LDGC (in the first query) is at least reduced by a factor of eight w.r.t. GRIS.

For the remaining nineteen queries the results are very similar (in practise no difference between GRIS and LDGC) and we can also note this in Figure 4.

We can justify the differences in time related to the first query considering that when cache expires GRIS does not restart provider until there is an incoming query that needs those information (this *lazy behavior* is an issue related to GRIS and not to LDAP protocol). Instead, in LDGC, providers are automatically restarted when information cache is expired; in this way updated information is always available for every incoming query.

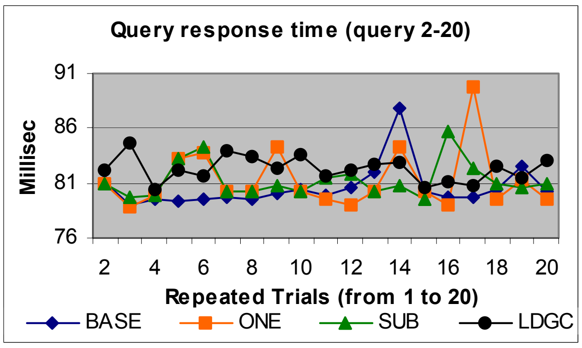

Figure 4. Query response time (repeated trials from 2 to 20)

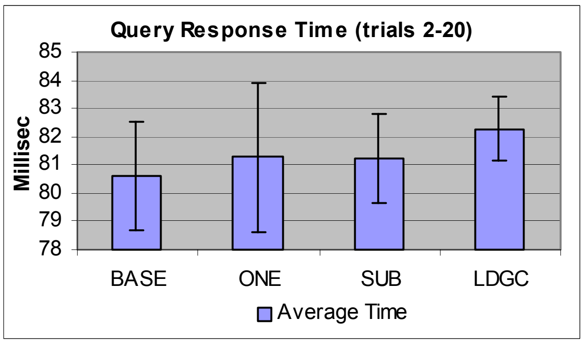

Figure 5. Query average response time and variance (repeated trials from 2 to 20)

Let us now consider Figure 5. In this graphs we reported average response time considering variance for query going from number 2 to number 20. There is no considerable difference between the two proposed approaches from the average time point of view.
Figure 4 showed that GRIS response times are less stable (more fluctuations) than the ones related to LDGC. Indeed variance in query response time in GRIS is higher than in LDGC (see Figure 5).
In this version no optimizations were introduced in the LDGC to speedup the output results. In the future, some caching mechanisms, as for instance some views, will be introduced to improve performance of our LDGC Information Service.

5. RELATIONAL INFORMATION SERVICE

The presented relational model for information handling has been used as starting point for the design of the iGrid Information Service developed within the European GridLab project, namely within work package 10 [14], leaded by the Center for Advanced Computational Technologies of the University of Lecce in Italy.
As shown in fig. 6, the iGrid distributed architecture is based on two kind of nodes, the iServe and the iStore GSI enabled web services. The iServe collects information related to a specific computational resource, while the iStore gathers information coming from registered iServes.
The current architecture allows iStores to register themselves to other iStores, thus creating distributed hierarchies, improving the fault tolerance of the entire system.

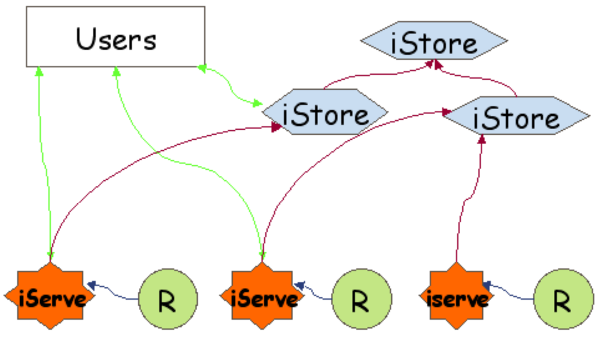

Figure 6. iGrid Architecture

The Information Service is based on a relational DBMS (PostgreSQL is currently used as back-end) and can handle information extracted directly from a computational resource (through a set of information providers), but also information directly supplied by users allowing users to store information. To date, iGrid Information System can handle information related to:

- ✓ *System*: belongs to this class, information like operative system, release version, machine architecture;
- ✓ *CPU*: for a CPU is extracted static information like model, vendor, version, clock speed; but also dynamic information like idle time, nice time, user time, system time, load;
- ✓ *Memory*: related to memory is available static information like amount of RAM, swap space size, and dynamic information like available memory space, available swap space;
- ✓ *File Systems*: static information is extracted as well as dynamic information; some examples include file system type, mount point, access rights, size, available space;
- ✓ *Network Interfaces*: information that belongs to this category include: network interface name, network address, network mask;
- ✓ *Local Resource Manager*: the information belonging to this category can classified further more in three different subclasses: information about queues, and information about jobs, and static information about Local Resource Management System (LRMS). Some example of information that can be extracted is LRMS type, LRMS name; queues name, status, number of CPU assigned to the queue, maximum number of jobs that can be in the queue, number of jobs queued, etc.; finally, jobs name, identifier, owner, status, submission time, etc. Currently only information providers for OpenPBS, and Globus Gatekepeer are available;
- ✓ *Certification Authorities*: information related to trusted certification authority such as subject name, serial number, expiration date, issuer, PK algorithm, etc.;

A set of information can be also supplied by user. The user supplied information are:

- ✓ *Firewall*: related to firewall we have information like the name of the machine where the firewall is installed on, the administrator name, the range of ports allowed to pass through the firewall;
- ✓ *Virtual Organization*: information related to VO can be used to automatically discover which resources belong to a given VO. In this category we have VO name, resource

type, a help desk phone number, help desk URL, the job manager, etc.;

- ✓ *Service and Web Service*: it is also possible to use the information service for service or web service discovery. In such a case information like service name, description, WSDL location, keyword are available;

The implementation includes system information providers outputting XML, while user information is directly supplied by the user simply calling a web service registration method.

iGrid uses a push model for data exchange: information extracted from resources are stored on the local DBMS, and periodically sent to registered iStores, while user supplied information is immediately stored on local DBMS and sent to registered iStores. Thus, an iStore has always fresh, updated information, and does not need to ask iServes for information. Moreover, each information is tagged with a time to live that allows iGrid to safely removes stale information from the DBMS as needed. Indeed, on each user lookup, data clean-up is performed before returning to the client the information requested. When iGrid starts, the entire DBMS is cleaned up. Thus the user will never see stale information.

Fault tolerance works as follows. In case of failure of an iStore, iServes remove temporarily the faulty iStore from their registration list. Periodically, the iStore list is updated by adding previously removed iStores (when iStores are available again). In this case, the local DBMS is dumped and immediately sent to newly added iStores.

The iGrid web service is based on the gSOAP toolkit with the GSI plugin and the GRelC libraries, it uses libxml2 library to parse XML documents and information providers use gtop library to extract the needed information from resources. Finally iGrid support TSL on back-end for binding to relational DBMS, supports GAS authorization service [15] for user authorization and Mercury logging service for user access and service usage monitoring [16].

We are also investigating a possible implementation of a peer to peer overlay network based on one of the current state of the art distributed hash table algorithms in order to improve iGrid scalability.

6. THE GRID RELATIONAL CATALOG PROJECT

Starting from the Dynamic Grid Catalog Information Service, we moved towards a more general purpose framework for adaptive data management in a grid environment, that is the Grid DataBase Management System [17,18]. In our definition, it is *"a system which dynamically, automatically and transparently reconfigures at runtime, components such as Data Resources according to the Grid state in order to maintain a desired performance level. It must offer an efficient, robust, intelligent, transparent, uniform access to Grid-Databases."*

The Grid Relational Catalog project [19] (developed at the CACT/ISUFI Laboratory of the University of Lecce), aims at providing a first implementation of the Grid-DBMS specification.

It provides a set of high level and grid-enabled data services such as for instance:

- ✓ Data Access Service (DAS): it provides a standard database access interface for relational and not-relational (i.e. textual database) data sources. It is based on the core DGC library and provides an extension of that set of APIs (currently the DAS libraries [20] contains more than 80 APIs whereas in the DGC library only 12 APIs were developed). Furthermore additional features (GridFTP protocol [21,22,23] and compression mechanisms) are also supported in the DAS to improve performance and speedup query submission and data retrieval processes [24]. A wider set of operations related to data manipulation (i.e. movefirst_record, movenext_record, find_records, etc.) is also available to support user applications development.
- ✓ Data Gather Service (DGS): it provides a data integration service, which gathers information coming from several DAS [25]; indeed it is placed among grid applications and the Data Access Services. It exploits the libxml2 [26] and the DAS libraries. Currently two versions of the DGS are already implemented (the GRelC Gather Service [27,28] – GGS - and the Enhanced GGS [29]). The EGGS exploits the Global-Global-DGC inquiry protocol (additional details about this protocol can be found in [4]) in order to improve efficiency and scalability.
- ✓ Dynamic Reconfiguration Service (DRS): a "scheduling-based" service which is responsible for automatically reconfiguring (that is, replicating, relocating and partitioning) data sources accordingly to host and data sources performance.
- ✓ Data Monitoring Service (DMS): a service which aims at monitoring the entire Grid (hosts and databases performance) in order to obtain information useful for making decision processes (see DRS);
- ✓ Data-Optimizer Service (DOS): a service which aims at optimizing the performance related to the data sources creating views, indexes and so on, based on previous statistics and performance threshold defined by the administrator.

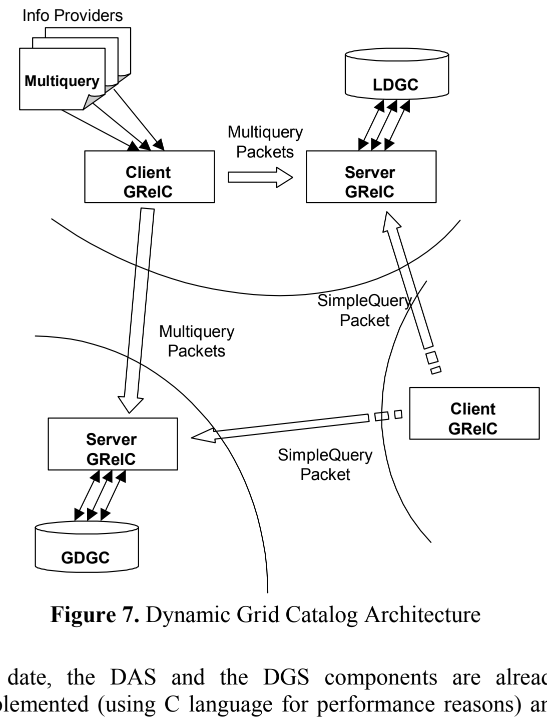

Figure 7. Dynamic Grid Catalog Architecture

To date, the DAS and the DGS components are already implemented (using C language for performance reasons) and used in several grid applications/projects related to Earth Observation System (Distributed EOS Information Service [29,30]), Healthcare (Virtual clinical folder on the Grid [27]), and Bioinformatics (ProgenGrid project [31,32]). Moreover core libraries (such as the grelc_multiquery_lib_v1.0 and

grelc_sdai_lib_v1.1) are also used in other projects such as for instance the iGrid Information Service (see previous section).
The DAS and DGS (both the client/server and the Web Service GSI enabled version) provide several features such as security (*authentication, authorization, delegation, data encryption, access control policy*), transparency (*supplying grid applications with dynamic binding to heterogeneous DBMSs – PostgreSQL [33], MySQL [34], Oracle [35], etc.*), efficiency (*high throughput, concurrent accesses, fault tolerance, reduced communication overhead*) and dynamicity (*dynamic mechanisms illustrated before*).
Currently, the DAS, provides full support for several kinds of not-traditional queries such as:

- *MultiQuery*: it represents a new mechanism useful to submit a huge amount of INSERT, DELETE and UPDATE queries in a single shot, reducing both the connection time and the interactions between client applications and DAS; this can lead to interesting improvements in the query submission process as reported in [36]. Moreover, jointly using compression mechanisms (zlib [37]) and GridFTP protocol we can strongly reduce the MultiQuery transfer time from client applications to DAS. The MultiQuery mechanism can be extensively used within whatever relational information service, because it represents an attractive and efficient data exchange format between Information Providers (IP) and Data Collectors Nodes (i.e. LDGC). The basic version of the DGC Relational Information Service, described in this paper, carried out this kind of query to push data from computational resources to LDGC and GDGC (see Figure 7)
- *WorkflowQuery*: multiple query with dependencies submitted to several data sources (the output of a query provides the input for the next one). We plan to use this kind of query into complex bioinformatics experiments that needs to combine elementary task (life science basic applications such as protein to protein comparison) into a workflow structure.
- *ActionQuery:* query jointly used with job submission on grid resources (the resultset of a query can be seen as the input for a program installed on a grid node);

As a future work, we plan to implement all of the services described before (i.e. DRS, DMS and DOS), moving towards a Grid Services architecture (Open Grid Service Architecture – OGSA [38] or the emerging WSRF [39]).
Several efforts will be addressed in the *dynamic mechanisms* described in the Grid-DBMS definition, in order to develop an efficient adaptive framework for data management in a grid environment.

7. CONCLUSIONS

In a grid environment information is a critical element. In this paper we presented a performance analysis between Grid Resource Information Service (GRIS) and Local Dynamic Grid Catalog (LDGC). Considering the same set of information in our tests the average queries response times were very similar when data were available in the cache; on the contrary we obtained different results with an empty cache (response time for LDGC was reduced by a factor of eight w.r.t. GRIS). Furthermore we obtained other interesting results related to stability (response time for LDGC was more stable then GRIS). We justified the different times related to the first query considering that when cache expires LDGC and GRIS manage information providers differently.
After the analysis of experimental results, we described the Relational Information Service *iGrid* (developed within the European GridLab project), which leverages the relational model for handling information into a Grid Information Service and the Grid Relational Catalog project (GRelC), which provides a set of data management, access and integration services in a grid environment.

8. REFERENCES

[1] Ian Foster, Carl Kesselman and Steve Tuecke. 2001. The anatomy of the grid: Enabling scalable virtual organizations. **International Journal of Supercomputer Applications**.

[2] The Globus Project, http://www.globus.org

[3] Monitoring Discovery Service, http://www.globus.org/mds/

[4] G.Aloisio, M.Cafaro, E.Blasi, I. Epicoco, S.Fiore, M.Mirto, Dynamic Grid Catalog Information Service, **Proceedings of the First European Across Grids Conference**, *February 13-14, 2003 Santiago de Compostela (Spain)*, Lecture Notes in Computer Science, Springer-Verlag, N. 2970, 2003,pp. 198-205

[5] Peter Dinda and Beth Plale. 2001. **A unified relational approach to grid information services**. Technical Report GWD-GIS-012-1,GGF. [http://www.cs.northwestern.edu/~urgis/gis012.pdf]

[6] W. Hoschek, G. McCance. 2001. **Grid Enabled Relational Database Middleware**, Informational Draft URL: [http://www.cs.northwestern.edu/~pdinda/relational-gis/hoschek_mccance_ggf3.pdf].

[7] B. Plale, P. Dinda, and G. von Laszewski. 2002. Key Concepts and Services of a Grid Information Service, **Proceedings of the 15th International Conference on Parallel and Distributed Computing Systems** (PDCS 2002)

[8] Relational Grid Information Services (RGIS-RG), http://www.gridforum.org/1_GIS/RDIS.htm

[9] Grid Forum. Information Systems and Performance area. http://www.gridforum.org/

[10] G. Aloisio, M. Cafaro, I. Epicoco, S. Fiore. 2002. **Analysis of the Globus Toolkit Grid Information Service**. Technical Report GridLab-10-D.1-0001-1.0. [http://www.gridlab.org/Resources/Deliverables/D10.1.pdf]

[11] S. Tuecke, 2001 **Grid Security Infrastructure (GSI) Roadmap**. Internet Draft. [www.gridforum.org/security/ggf1_2001-03/drafts/draft-ggf-gsi-roadmap-02.pdf]

[12] Heinz Johner, Larry Brown, Franz-Stefan Hinner, Wolfgang Reis, Johan Westman - **Understanding LDAP** [http://www.redbooks.ibm.com/pubs/pdfs/redbooks/sg244986.pdf]

[13] Eric K. Clemons. **Principles of Database Design**, Vol 1. Prentice Hall, 1985.

[14] http://www.gridlab.org/WorkPackages/wp-10/index.html

[15] http://www.gridlab.org/WorkPackages/wp-6/index.html

[16] http://www.gridlab.org/WorkPackages/wp-11/index.html

[17] G. Aloisio, M. Cafaro, S. Fiore, M. Mirto, The GRelC Project: Towards GRID-DBMS, **Proceedings of Parallel and Distributed Computing and Networks** (PDCN) – IASTED, February 17 to 19, 2004, Innsbruck, Austria.

[18] J. S. A, Gounaris, P. Watson, N. W. Paton, A.A.A.

Fernandes, and R. Sakellariou. Distributed query processing on the grid. In **Proceedings of the 3rd International Workshop on Grid Computing** (GRID 2002),. LNCS 2536, Springer-Verlag, 2002, pages 279-290

[19] The GRelC project. [http://gandalf.unile.it].

[20] G. Aloisio, M. Cafaro, S. Fiore, M. Mirto, The GRelC Library: A Basic Pillar in the Grid Relational Catalog Architecture, **Proceedings of Information Technology Coding and Computing** (ITCC), April 5 to 7, 2004, Las Vegas, Nevada, Volume I, pp.372-376.

[21] Grid Forum **GridFTP Introduction** [http://www.sdsc.edu/GridForum/RemoteData/Papers/gridftp_intro_gf5.pdf]

[22] Grid Forum **GridFTP Specification** DRAFT [http://www.sdsc.edu/GridForum/RemoteData/Papers/gridftp_spec_gf5.pdf]

[23] **GridFTP Protocol**. URL: [http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf]

[24] G. Aloisio, M. Cafaro, S. Fiore, M. Mirto, Advanced Delivery Mechanisms in the GRelC Project to appear in the **Proceeding of 2nd International Workshop on Middleware for Grid Computing** (MGC 2004), October 18, 2004, Toronto, Ontario (Canada).

[25] M. Ozsu and P. Valduriez. **Principles of Distributed Database Systems**. 2nd edition. Prentice Hall, Upper Saddle River, NJ, USA, 1999.

[26] Extensible Markup Language (XML), URL:[http://www.w3.org/XML/].

[27] G. Aloisio, M. Cafaro, E. Blasi, S. Fiore, M. Mirto, A Virtual Clinical Folder on the Grid, in **Proceedings of SCI2004**, 18 - 21 July 2004, Orlando, Florida , USA

[28] M. Mirto, G. Aloisio, M. Cafaro, S. Fiore, A Gather Service in a Health Grid Environment, CD-Rom of **Medicon and Health Telematics 2004**, IFMBE Proceedings, Volume 6, July 31 – August 05, Island of Ischia, Naples, Italy.

[29] G. Aloisio, M. Cafaro, S. Fiore, G. Quarta, A Grid-Based Architecture for Earth Observation Data Access, submitted to the **20th Annual ACM Symposium on Applied Computing (SAC-2005)**, Santa Fe, New Mexico, March 13 -17, 2005, Mexico.

[30] igeo project. [http://leonardo.unile.it/igeo]

[31] G. Aloisio, M. Cafaro, S. Fiore, M. Mirto, ProGenGrid: A Grid Framework for Bioinformatics, to appear in the **Proceeding of International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics** (CIBB 2004), September 14-15 2004, Perugia, Italy.

[32] G. Aloisio, M. Cafaro, S. Fiore, M. Mirto, Bioinformatics Data Access Service in the ProGenGrid System to appear in the **Proceeding of First International Workshop on Grid Computing and its Application to Data Analysis** (GADA 2004), October 25-29, Larnaca, Cyprus, Greece.

[33] PostgreSQL, URL: [http://www.postgresql.org/]

[34] MySQL, URL: [www.mysql.com]

[35] Oracle Grid Computing Technologies URL: [http://otn.oracle.com/products/oracle9i/grid_computing/index.html].

[36] G. Aloisio, M. Cafaro, S. Fiore, M. Mirto, Early Experiences with the GRelC Library, **Journal of Digital Information Management**, Digital Information Research Foundation (DIRF) Press. Vol. 2, No. 2, June 2004, pp 54-60.

[37] Zlib, URL: [http://www.gzip.org/zlib]

[38] Foster, I., Kesselman, C., Nick, J., & Tuecke, S. (2002). **The Physiology of the Grid: An Open Grid Services Architecture for Distributed System Integration**. Technical Report for the Globus project. URL: http://www.globus.org/-research/papers/ogsa.pdf.

[39] **WS Resource Framework** (WSRF). URL: [http://www.globus.org/wsrf/].