Gredia Middleware Architecture *

Ioannis Konstantinou, Katerina Doka, Athanasia Asiki, Antonis Zissimos
and Nectarios Koziris

National Technical University of Athens
School of Electrical and Computer Engineering
Computing Systems Laboratory
Zografou Campus, Zografou 15773, Greece
email: {ikons, katerina, nasia, azisi, nkoziris}@cslab.ece.ntua.gr

Abstract

In this paper, we present **RDLS (Rich Data Location Services)**, a grid middleware for data management designed to meet the requirements posed by the *GREDIA* project [1]. The main purpose of the described Middleware components is to provide the prerequisite services for storage, search and retrieval of annotated, rich media content in a large-scale distributed environment. These services can be efficiently integrated in existing Grid middlewares, such as Globus Toolkit 4 [5], to manipulate rich media content. The users of *GREDIA* platform will be able to upload their files in the distributed *GREDIA* repository and to perform advanced searches in the annotations of the stored content. A Peer-to-Peer overlay will be implemented to store the provided annotations. Every node connected to the platform will be able to act not only as a consumer of storage resources, by means of searching existent files, but also as a provider of storage capacity and data services.

1 Introduction

The explosive evolution of networking infrastructures along with the growing adoption of grid computing initiatives in resource sharing create attractive perspectives for the deployment of large-scale, distributed computing and storage systems. On the other hand, as far as data management is concerned, *Peer-to-Peer* (P2P) algorithms have emerged as a promising solution, offering incontestable advantages in terms of scalability, fault-tolerance and the ability to adapt in dynamic node arrivals and departures [2]. A distributed facility incorporating practices from both areas has been envisioned by numerous related research initiatives.

The *GREDIA* middleware architecture has two significant characteristics: It is **modular**, namely it has well defined different components that interact with each other in a clear way and it is **standardized**, that is, it complies with well defined and widely accepted by the Grid community standards.

*The described work is partly supported by the European Commission through the FP6 IST Framework Programme (FP6-34363 [1]).

2 Interoperability

The development of *RDLS* is based in Open Standards, to facilitate interoperability with other currently used systems. The existing services layer exposes the functionality of the data architecture to other components of GREDIA platform. Considering this fact, this layer should use standard protocols, in order to communicate with other services. Well defined interfaces should be supported to enable the integration of services from different subsystems. The most appropriate technology to achieve interoperability is the *Open Grid Services Architecture*(*hence OGSA*), as it was suggested by Foster et al. in [6] and was later enhanced, in [4]. The *OGSA* architecture is based upon another standardization protocol adopted by the majority of both the academic and the industrial community, the *Web Services Architecture* [3].

The mechanics of a Web Service are described in a *Web Service Descriptor*. The WSD is a machine-aware specification of the Web service's interface, written in *WSDL (Web Services Description Language)*. In a WSD the web service's interface is documented in a machine readable manner, using XML tags to create the WSDL document. Services use each other's WSD to discover the appropriate interfaces in order to communicate.

In the *Web Services Resource Framework* [4], the appropriate extensions are presented, so as to apply the notion of state in the otherwise stateless Web Services. As of this, WSRF services, in contrast to pure WS services, have **persistent state** that remains consistent during numerous transactions with other services. During the initialization of a WSRF service, a unique identifier is created and passed to the client. Using this identifier, the client can query the state of the service at anytime, and this applies to all other clients that are aware of the specific unique identifier.

3 Components Interaction

The *RDLS* components interact with each other to offer a set of core data services to other GREDIA Components. In Figure 1, the detailed stucture of the proposed architecture is presented.

3.1 Data Service

The *Data service* is the service for manipulating data items. Its role is double:

- It receives a client's request for upload of a data item and stores it through a web based application. A unique identifier (CurrentID) is assigned to the data item, which is then uploaded to the Storage overlay.
- It retrieves a file from the Storage overlay and returns a data stream to the search clients, which are then able to download the data item using the GridTorrent protocol [10].

This service is executed in conjunction with the Metadata service. The unique identifier, called *(CurrentID)* is used to map the metadata file to the

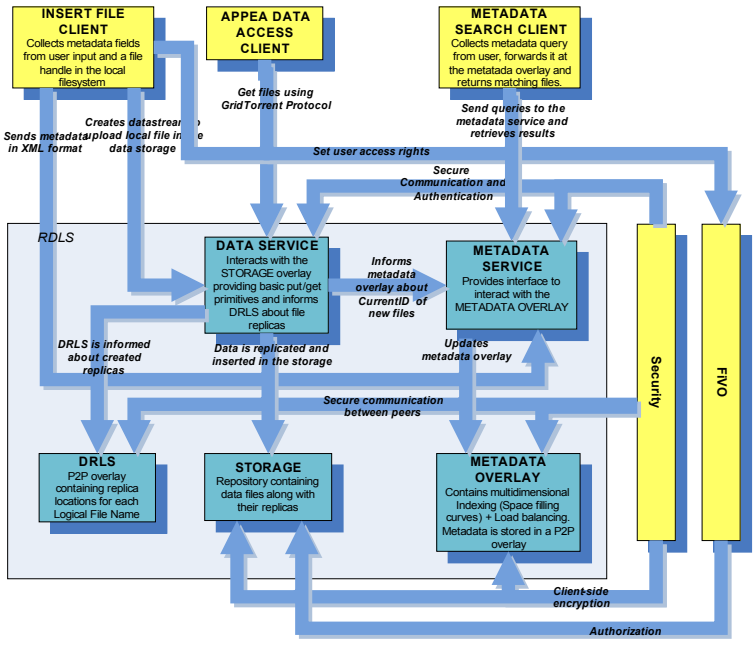

Fig. 1: GREDIA Middleware Architecture

corresponding date file stored in the Storage overlay. After the initial invocation of the Data Service for the upload of a file, the specific file is available to other users of the GREDIA platform. This service is also responsible for the data retrieval from the physical locations of data replicas. The mappings of CurrentIDs to physical locations are stored in a distributed catalogue implemented in the DRLS overlay. It is obvious that a user needs to be authenticated in order to be granted access to the DataService.

3.2 Metadata Service

The *Metadata Service* is responsible for the assignment of metadata files, describing a data item, to the peers of the Metadata Overlay. It also processes and answers queries initiated either from the Appea Data Client, or by a separate, web-based metadata search client. This is achieved by forwarding the queries to the Metadata overlay in order to retrieve relevant files, according to their metadata description. In case of data insertion, the Metadata service receives the CurrentID of the data item produced by the DataService. The CurrentID along with the object's metadata are used to calculate the exact metadataId using an algorithm to reduce dimensions [9],[7]. The metadataId is used to find the node in the Metadata overlay responsible for storing the data item's XML description. In the case of data search, it forwards the XML query to the Meta-

data Overlay, which should return the XML metadata description of the data items satisfying the query. This service also prerequisites user authentication and secure communication.

3.3 Metadata Overlay

The Metadata overlay is a **P2P DHT-based overlay** [8] holding metadata files. Metadata files are assigned to a corresponding node according to its metadata values. To ensure system scalability and to achieve minimization of response time and communication overhead implied by the searching procedure, multidimensional indexing techniques based on **Space Filling Curves** [9],[7] are exploited during the implementation of the Metadata Overlay. All communication among peers should be secure, and only authorized users should be granted access.

3.4 DRLS (Distributed Replica Location Service)

DRLS is a **Distributed Replica Location Service (RLS)** and consists of a P2P overlay that contains the mappings of LFNs to PFNs. The DRLS overlay returns all replica locations for a specific LFN. It interacts with the Data Service, namely DRLS provides the Data Service with PFNs indicating replica locations of requested files and the Data Service informs DRLS about newly created replicas. All communication between peers of the P2P overlay should be secure.

3.5 Storage

This is the actual repository of data items. The Storage component receives the data item along with its replicas from the Data Service and stores them to the physical nodes. Authorization is needed to access a data item.

4 Middleware Services API

In this section, we will elaborate on the aforementioned services, describing in detail the exact way to invoke them. In more details, the middleware provides the following services:

4.1 Name: saveMetadata

Purpose: It makes the metadata entered by a user available to other peers, which can locate them with the search service later.

Description: During the insertion of a new file in the GREDIA repository, a user fills in a form with metadata to describe the actual file. The metadata file is uploaded at the Metadata overlay and it is indexed for fast retrieval. A unique identifier is needed to describe each file. This identifier is the system-generated CurrentID. The same identifier is used both in the storage and replica location

overlay. The saveMetadata service is invoked along with the saveData service, to ensure that the LFN is the same both in the metadata and storage overlay.

Parameters: XML Metadata: XML document

IN: An XML document containing all the keywords that a user has inserted during the upload procedure. The structure of this XML is described by an appropriate XSD file.

Returns: If the file is successfully uploaded, the saveMetadata service returns an "OK" signal. Otherwise, it returns an error code describing the error cause (for instance, if the user has no write permission).

4.2 Name: Search

Purpose: It sends a query to the Metadata overlay to retrieve relevant files.

Description: GREDIA users upload files in a large-scale geographically distributed repository. The uploaded files are annotated according to a predefined metadata schema. These annotations are stored in the Metadata overlay and properly indexed to avoid latencies and to ensure redundancy. The indexing procedure is transparent to applications; they are able to send their queries using only this service. The output of this service is an XML document containing the LFNs corresponding to files answering the imposed query.

Parameters: XMLQuery XML document

IN: An XML document containing the user queries.

Returns: An XML file, which contains a list of the metadata XMLs, which are returned as results.

4.3 Name: saveData

Purpose: It stores a local file in the Storage overlay.

Description: Users are able to share their files with other users participating in the GREDIA platform. The role of this service is to upload a file at the storage overlay and assign to it a unique identifier. This service is executed in conjunction with the saveMetadata service, as the unique identifier must be the same for the metadata and the storage overlay. After the execution of the service, the file is available for search and retrieval by users with the required permissions.

Parameters:

Current ID

IN: The desired unique ID that identifies a specific version of the file in the storage overlay (in the overlay different versions of files with the same LFN will exist, as there is support for versioning. As of this, the distinguishing characteristic of the different file versions is the Current ID)

Datastream

IN: A pointer to a data stream in the local repository of the user

Returns: Returns an "OK" signal if the file is successfully uploaded, otherwise it returns an error code describing the error cause (for instance, the LFN already exists, or the user may not has write permission).

4.4 Name: getData

Purpose: Retrieves a file from the storage overlay

Description: This service returns a data stream to the application Layer of the file it wants to receive.

Parameters: Current ID

IN: The desired unique ID that identifies a specific version of the file in the storage overlay (Multiple versions of the same file have the same LFN and are distinguished by the different Current IDs. In this way, a versioning scheme is provided.)

Returns: A pointer to a data stream for the requested file in the storage overlay

5 Conclusion

Our goal is to develop extensions to the existing grid middleware [5] to support management of Rich Media Content, which is currently missing. Using OGSA and open source code, we are developping generic data middleware that can be reused by other grid platforms, facilitating interoperability between different grid systems.

References

1. Gredia.eu. http://www.gredia.eu/?Page=home.
2. H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking up data in p2p systems. *Commun. ACM*, 46(2):43–48, 2003.
3. D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web services architecture. *W3C Working Group Note*, 11:2005–1, 2004.
4. K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke. From open grid services infrastructure to ws-resource framework: Refactoring & evolution. *Global Grid Forum Draft Recommendation, May*, 2004.
5. I. Foster. Globus toolkit version 4: Software for service-oriented systems. *Network And Parallel Computing: IFIP International Conference, NPC 2005, Beijing, China, November 30-December 3, 2005: Proceedings*, 2005.
6. I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The physiology of the grid. *Grid Computing: Making the Global Infrastructure a Reality*, pages 217–249, 2003.
7. H. V. Jagadish. Linear clustering of objects with multiple attributes. *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, pages 332–342, 1990.
8. P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 258:263, 2002.
9. B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz. Analysis of the clustering properties of hilbert space-filling curve. *Analysis*, 13:124–141, 2001.

10. A. Zissimos, K. Doka, A. Chazapis, and N. Koziris. Gridtorrent: Optimizing data transfers in the grid with collaborative sharing. In *11th Panhellenic Conference on Informatics (PCI2007)*, Patras,Greece, May 2007.

Meaning of Middleware. Middleware is also known as plumbing because itâ€™s both sides are connected with different applications, and it helps to transfer data in both side. Middleware was developed in 1980s for making linking with new designed applications to older legacy systems. Now, it is used in various areas such as web servers, application servers, content management systems, and other tools which help to design applications and delivery. Gredia Middleware Architecture â^— Ioannis Konstantinou, Katerina Doka, Athanasia Asiki, Antonis Zissimos and Nectarios Koziris National Technical University of Athens School ofÂ designed to meet the require- ments posed by the GREDIA project [1]. The main purpose of the de- scribed Middleware components is to provide the prerequisite services for storage, search and retrieval of annotated, rich media content in a large-scale distributed environment. Making own middlewareâ€™sÂ¶. Coming soonâ€¦ Available middlewareâ€™sÂ¶. Coming soonâ€¦ Next Previous. Â© Copyright 2019, Illemius / Alex Root Junior Revision 9b9c1b08. Built with Sphinx using a theme provided by Read the Docs. Read the Docs v: latest. Versions. Middleware in the context of distributed applications is software that provides services beyond those provided by the operating system to enable the various components of a distributed system to communicate and manage data. Middleware supports and simplifies complex distributed applications. It includes web servers, application servers, messaging and similar tools that support application development and delivery. Middleware is especially integral to modern information technology based on XML, SOAP... In this page we present installation instructions for the gredia middleware platform. Specifically, we present instructions for the following components: RDLS, SFC, Gridtorrent. RDLS. RDLS (Rich Data Location Services) is the layer that exposes the middleware's functionality through standard Web Services functionality. In the following, we give installation instructions of how to setup a gredia middleware server and how to install the client software and access the functionality of the installed services. Server setup.