MetaData for Efficient, Secure and Extensible Access to Data in a Medical Grid

Jean-Marc Pierson, Ludwig Seitz, Hector Duque, Johan Montagnat

MetaData for Efficient, Secure and Extensible Access to Data in a Medical Grid

Jean-Marc Pierson[1] Ludwig Seitz[1] Hector Duque[1,2] Johan Montagnat[2]
[1]LIRIS, CNRS FRE 2672
[2]CREATIS, CNRS UMR 5515, INSERM U630
INSA de Lyon, bât B. Pascal, 7, av. Jean Capelle, 69621 Villeurbanne cedex, FRANCE
{ludwig.seitz, jean-marc.pierson}@liris.cnrs.fr, {johan,duque}@creatis.insa-lyon.fr

***Abstract*— In this paper we present the metadata usage in a medical imaging project grid. Metadata represent data about the data: In our case, the data are medical images and the metadata store relative information on the patient and hospital records, or even data about the image algorithms used in our application platform. Metadata are either static or dynamically constructed after computations on data. We show how the metadata is used, produced and stored to provide a secure and efficient access to medical data (and metadata) through a dedicated architecture. Experiments include times to access data and to secure the transactions.**

Keywords: metadata management, access control, medical grids

I. Introduction

In the past years, Grid Computing has emerged as a new tool to accommodate the needs of process intensive applications (such as weather forecast, nuclear simulation, ...) and the needs of scientists around the world. Focus has long been put on the efficient use of computing resources, in terms of scheduling, resources discovery and usage, in projects like Globus [8] or Legion [10]. Interest in the data being manipulated by grids has grown only in the very last years, when the amount of data used or produced in grid applications started to become a real problem while no (or almost no) research was driven for their management. The DataGrid [6] project for instance was one of the first project to focus also on data.

On the other hand, Grids are becoming popular among the Information Systems community for their ability to handle tons of data. From an IS point of view, application data is not considered as raw data like in many grid applications but rather as semantically rich data or data enriched by the use of metadata. The current interest in semantic grids and metadata management at the Global Grid Forum is an indicator of this trend.

Finally, metadata might be dynamic. Indeed, some computation made on the grids on data may be stored for future use and becomes not only data, but also metadata on the data itself. The link between original data and newly produced data must somehow exist. We believe that the dynamic metadata will be more and more numerous, thus motivating this work on their management.

This work is partly supported by the Région Rhône-Alpes project RAGTIME, the French ministry of research ACI-GRID program, and the ECOS Nord Committee (action C03S02).

Ability to handle raw data and semantically enriched data in the same architecture is mandatory for future grid expansion and usage. Medical grids provide a good field of experiment for data and metadata management. Indeed, data range from raw images from an acquisition device to patient related data, with a need for privacy protection and hybrid requests on images and patient data.

The rest of the document is organized as follows : Section II describes the motivation for metadata in the medical field and section III gives a short overview of our data and metadata access architecture. Section IV provides details on the usage of static and dynamic metadata in our system, for access control and efficient access. Section V deals with implementation and experiments issues. Section VI mentions related works and discusses our metadata management while section VII concludes.

II. Motivation for Medical Metadata

Medical images have become a key investigation tool for medical diagnosis and pathology follow-ups. Digital imaging is becoming the standard for all image acquisition devices and with the generalization of digital acquisition, there is an increasing need for data storage and retrieval. The DICOM (Digital Image and COmmunication in Medicine) has recently emerged as the standard for image storage. DICOM describes an image format, a communication protocol between an image server and its clients, and other image related capabilities. On top of such a standard, PACS (Picture Archiving and Communication Systems) are deployed to manage data storage and data flow inside hospitals.

However, medical images by themselves are not sufficient for most medical applications. A physician is not analyzing images but he needs to interpret an image or a set of images in a medical context. The image content is only relevant when considering the patient age and sex, the medical record for this patient, sociological and environmental considerations, etc. Beyond simple diagnosis, many other medical applications are concerned with the data semantics and require rich metadata content. For instance, epidemiology requires the study of large data sets and the search of similarities between medical cases. Physicians are often interested in looking for medical cases similar to the one they are studying. A case may be identified as “similar” because the image contents are similar

but this is often not sufficient to discriminate between a set of acquisitions. There is a need to take into account metadata on the medical case and results of computations done on the images in the similarity criterion. Therefore, medical metadata carrying additional information on the images are mandatory.

DICOM images indeed contain some acquisition-related metadata in the image header. However, this in-file metadata is often incomplete and not practical for data search and query. Therefore, DICOM servers usually extract the in-file metadata and store it in databases. In addition to PACS, hospitals have a need for RIS (Radiological Information Systems). The PACS archives the images and allows image transfers. The RIS contains full medical records: image-related metadata and additional information on the patient history, pathology follow-up, etc. There exists no open standards for the data structure and the communication between the services in this architecture. Moreover, they are usually designed to handle information inside an hospital but there is no system taking into account larger data sets nor the integration with an external component such as a computation/storage grid.

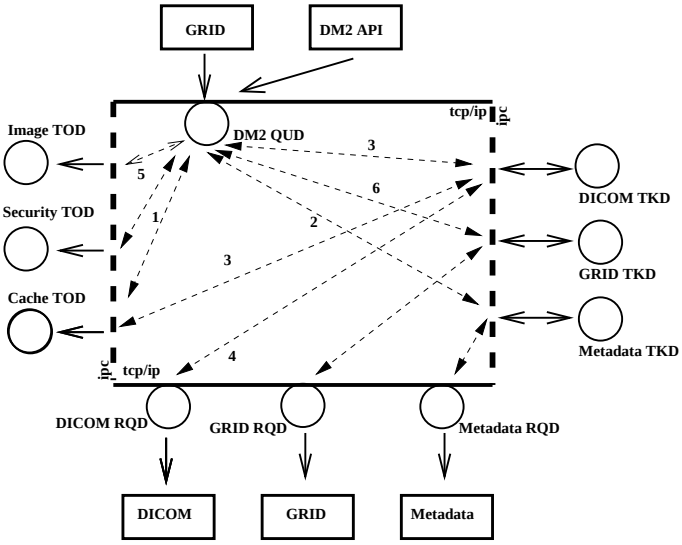

Fig. 1. DM2 architecture in use

III. Overview of the access architecture

To respond the medical data management requirements we propose a *Distributed Medical Data Manager* (DM^2). The DM^2 is designed as a complex system involving multiple grid services and several interacting processes geographically distributed over a heterogeneous environment. It also provides an access point to the grid services as well as an intermediate between the grid and a set of trusted medical sites. To tackle the DM^2 complexity we chose to first propose an architecture *(Distributed System Engines, DSE [5])* and then to implement our system as one possible instance of this architecture.

DSE services are composed of a set of independent processes which interact by exchanging messages. The architecture increases in semantic significance through five layers. The lowest level, DSE^0, is the message passing level enabling inter processes communications. The DSE^1 level brings atomic operations (*transactions*) to process complex requests composed of many messages. It offers the ACID properties: Atomicity, Consistency, Isolation and Durability. The top layers deal with distribution over several engines (DSE^2), offer a programming API (DSE^3) and a user interface (DSE^4).

The DM^2 system is a particular instance of this distributed architecture. DM^2 (see figure 1) uses different *internal tools (TOol Drivers)* : *CACHE* for improving the latency of accessing images, *SECURITY* for access control over a sequence of images, *IMAGELIB* for implementing operations (concatenation, format) over the images. A DM^2 server accesses external services such as *DICOM* Storage Service Class Providers [1] (*DICOM* TasK Driver and ReQuest Driver) and *MYSQL* to access multiple SQL metadata databases (*METADATA* TKD and RQD) . The *GRID* RQD submits jobs to MicroGrid (a Computing Grid developed in our laboratories).

As an example, the DM^2 engine is requested to execute an hybrid query, ie. find out similar images between an image database and a reference image. Such a request needs computation of similarity measures between the reference image and each image. Figure 1 details the operation; on top, the grid middleware triggers a DM^2 hybrid query (as an XML message) to get an image: (1) the engine first asks for access authorization *(SECURITY TOD)* and for image availability *(CACHE TOD)*. (2) If access is granted and image not available in cache, it accesses the database (*metadata TKD*) to locate the DICOM files from which the image must be assembled. (3) The *cache TOD* is requested again (the image might not be in the cache while some files might be). (4) Assuming the cache does not contain the requested file, it should be copied from the DICOM server. The DM^2 requests the DICOM server through the *DICOM RQD* and retrieves in parallel a set of DICOM slices that are assembled onto scratch space to produce the 3D image requested. (5) The DICOM files are assembled into a 3D image using an *image TOD*. (6) Finally, the image is stored into the cache and returned to the grid. This example is the one used in the experiment section.

IV. Metadata in use

Beyond simple patient-related metadata (age, sex, etc), the metadata sould also include:

- image-related metadata: image dimensions, voxels size, encoding, etc.
- acquisition-related metadata: acquisition device used, parameters set for the acquisition, acquisition date, etc.
- hospital-related metadata: radiology department responsible for this acquisition, radiologist, etc.
- medical record: anteriority, miscellaneous information explaining how to interpret this image, etc.

In addition to these medical metadata, external information is needed for computation-related maters. First, medical data are sensitive and should not be accessible by unauthorized users. In a grid computing context, data are likely to be transported between sites. They should no be readable by any third party spying the network communications. Second, it is often needed to track back data in order to assemble the

medical history of a patient. When producing a processed image, it is important to know the original data used for the creation of the processed image, the algorithm used and its parameter settings. Third, metadata can be used for data queries and computation optimizations. Computation on large images are costly and data retrieval in large image databases may represent untractable computation if image analysis is needed and images have not been properly indexed. The information related metadata therefore include:

- security-related metadata: authorization, encryption keys, data access logging, etc.
- history-related metadata: image sources, algorithms, parameters, etc.
- optimization-related metadata: image index, query caching, processing caching, etc.

As can be seen, some metadata are directly attached to the image, while other are related to the hospital or the patient. The metadata structure should therefore reflect these relations between images and metadata. Some metadata is *static*: it is either administrative information external to the image (*e.g.* patient metadata) or bound to the image (*e.g.* image metadata) with the same access pattern, same lifetime, etc. Other metadata is *dynamically* generated during computations. We can therefore classify the metadata:

- Static metadata.
  - External metadata: patient-related, hospital-related, medical record, security-related
  - Bound metadata: image-related, acquisition-related, security-related
- Dynamic metadata: history-related, optimization-related

Metadata is often very sensitive, even more than the image content itself: it contains all necessary information to identify patients and the security elements such as access control information and encryption keys. Most metadata can therefore only be stored on trusted and secured sites where administrators are accredited to manage such personnal data. Precise metadata access policies must be enforced.

As stated above, an important feature of a medical information system is its ability to retrieve relevant data for a given application. Data may be selected on the image content (by processing) or by taking into account its semantics (the metadata). Often both are needed at the same time. We refer to *hybrid* queries to designate queries of the information system that involve both selection on data content and metadata. Given the processing cost of image analysis algorithms, some computation results may be stored as new dynamic metadata bound to the images in order to optimize future computations. These new metadata become image index.

A. Metadata for access control

Our proposed access control system requires some ammount of metadata too. It uses certificates to store and transfer permissions as presented in [14]. This section describes and quantifies the metadata required by this service.

The data upon which we collect and store metadata is divided into files. Please note that in this context the term *file* designates a medical record and not a single physical file.

metadata	type	storage location	device	size
file-id to physical file	bound	storage-server	DB	<100 bytes
SOA to file-id	bound	storage-server	DB	<1 kbyte
file-id to file-set-id	bound	storage-server	DB	<100 bytes
CA certificate	external	storage-server	File	<1 kbyte
File transf. prog	bound	storage-server	File	varying
AC	external	storage-server, user	DB	<1 kbyte
AC revokation	external	storage-server	DB	<1 kbyte
Authentication certif.	external	user	File	<1 kbyte
Key-share	bound	third party	DB	<100 bytes

TABLE I
SUMMARY OF ACCESS CONTROL RELATED METADATA

We start by describing the metadata on the storage-server that also holds the files. The initial storage-servers in our case are the hospitals, but replica of the data might migrate secured on the Grid for performance issues.

- A unique identifier for each file (file-id). Since the access control system will span the entire grid environment, the file-id is unique over the whole grid.
- A unique user identifier (user-id) of a single source of authority (SOA) for each file. This SOA can issue access permissions and appoint secondary sources of authority for that file. We use RSA-public keys as user identifiers.
- If the file is an element in a file-set, a relation file-id to file-set-id is stored (one for each file-set).
- The certificate(s) of one or more trusted certification authorities (X.509 standard) that will allow to authenticate issuers of requests and to verify signatures from SOAs through a public key infrastructure (PKI).
- Should the file undergo some transformation before access (typically an anonymization) or should some sort of logfile be written for every access (for accounting and data traceability) the programs that perform those operations must be stored on the storage-server too.
- A certificate repository for external access. This enables SOAs to provide permissions to users that are not currently connected or to provide generic permissions to user groups.
- A collection of revoked certificates.

Other metadata is stored with the users using the grid services to access files. These are:

- Their authentication certificates (standard X.509 certificates, size $<$ 1 kbyte).
- Their attribute certificates. There are certificates granting them access rights either to files or to file-sets (size $<$2 kbytes).

Finally, we have metadata stored with a third party. Since files on the Grid are likely to be stored encrypted (using the system proposed in [13]), the key-shares that enable users to reconstruct the decryption keys will be stored on keyservers, that are neither colocated with the user nor the storage-server. These keyshares have a size of under 1kbyte too.

Table I summarizes all access control related metadata together with its type, storage location, storage device and approximate size.

The metadata thus provided is used to support a role-based access control service (see [7] for further details on role based

Metadata	column examples	type	storage location	size
image	storage server, image size	bound	dm2 client	284 bytes
medical	hospital, acq. date	external	storage-server	782 bytes
patient	patient id, patient name	external	storage-server	264 bytes
DICOM	dm2id, DICOM ID	bound	storage-server	260 bytes
log	source image, algorithm	dynamic	dm2 client	192 bytes
cache	dm2id, storage server	dynamic	dm2 client	128 bytes

TABLE II
SUMMARY OF MEDICAL METADATA

access control). This service allows to group users into roles, and files into sets. Users or roles can be assigned permissions to access files or file-sets. Additionally users or roles can be assigned to be members in other roles, thus creating the possibility for a hierarchic role structure. Such permissions are stored in attribute certificates that are signed by the concerned SOA or by a person that has been delegated to issue such permission by the SOA. The SOA for files is stored in the SOA to file-id relation. SOAs for file-sets and roles are encoded in the respective file-set or role identifiers. Thus a storage server can verify an access request solely based on his metadata and on the ACs the user supplies with his request without the need to contact a third party.

B. Metadata for efficient access

Once the authorization being granted to an user for accessing a data, the DM^2 architecture effectively retrieves the data itself. Two kind of queries are possible in the system : basic and hybrid queries.

Basic queries involve metadata such as the file names, and retrieve the set of files concerned by the query. This first set relies on static metadata hold on a metadata database that links each file with a relevant set of metadata, making the indexation of the images easy. Metadata about a DICOM file include basic information such as the size, the resolution, the acquisition date, etc, but also a $dm2ID$ identifier that links this file to the image file (fileid, introduced in section IV-A). Imagine a query addressing an image X of patient P. The system will find all the images of patient P, then it must select the image X, and also retrieve the relationship between the logical name X of the image, and the set of physical files constituting the image X. It must also give their respective locations in the grid, in order to be able to provide access to all needed files.

Hybrid queries associate a computation using the grid on some images, as well as a simple query. Imagine the user presents a source image to the system, and he wants to find all patients aged between 40 and 60, having some images similar to the source (the similarity measure resulting from some image analysis algorithm). The complexity of the hybrid query appears here clearly : First, the system must select the patients in the given age interval, then it must start the calculation of some potentially computation expensive similarity algorithm on the grid, get the results and compare these to extract the interesting cases.

Moreover, once the computation executed for one particular image, the result of the algorithm (that might be either a single number for a similarity measure -for the given example- or any other document like an image or a set of images in other more complicated algorithm) might also be stored and the link with the original source image have to be preserved. For each such dynamic new metadata, we generate an entry in the metadata database linking the source image, the result, and the algorithm being used. This allows again traceability as well as optimization. Indeed next time the same algorithm will have to be executed on the same image, the cached dynamic metadata (the result) will be retrieved without time consumption. From the previous example, if the user wants to narrow his search for patients between 45 and 55, it will not be necessary to restart the whole process but the cache will be used instead.

V. IMPLEMENTATION AND EXPERIMENTS

A. Implementation

The implementation of the system has been done in the framework of the European DataGrid, but the $DM2$ architecture has been developed without much connections to this grid, and only little adaptation is necessary to interface with other grids.

Originally devoted to use Spitfire [3] for the metadata database, which offers a secure layer above a MySql server, the team used a simple MySql server for the sake of simplicity and performance evaluation.

All the software has been implemented in C++ and the core API of the architecture, as well as description of the metadata and databases of the $DM2$ system can be retrieved from $http : //www.creatis.insa-lyon.fr/\ duque$.

B. Experiments

We have made experiments stressing a server engine which must access metadata before transferring a whole sequence of images (i.e. a set of Dicom files). We have installed locally a database having metadata (about localization).

For a set of images; we have installed also 8 client engines which simulate up to 8 hospitals where real raw data is stored. The experiment showed the response time when the server engine receives at the same time one hybrid query for each client engine, which means solving 8 hybrid queries in parallel. Those 8 hybrid queries at the same time mean querying metadata for those images and transferring 80 (8*10) Dicom files in parallel (since 10 files compose one image). The response is less than 3 seconds (2.8) for the first two queries, about 3.4 for the third one, and so on. the whole set of queries finishes at 4.5 seconds. In other words, one query takes about 3 seconds, but the set of 8 queries in parallel takes 4.5 seconds. The metadata access for the hybrid query is 0.35 seconds for each query.

We have also made some experiments for the access control (that has been excluded from previous experiment to determine its own cost), done via a tool driver. For each image, the time to compute the authorization has been measured on a average of 0.065s, and always below 0.1s. This time is composed of the time to read and parse the request (which had 3 access certificates), the time to access the metadata database twice (one to determine the administrator of the data, one to verify

that the the file is part of a set), and finally to compute the authorization decision based on these information.

```
This means :
tAC= 0.1 second        (time for access control)
tMD= 0.35 second       (time for metadata)
tD = 2.8-0.35 = 2.45 s (time for accessing data)
```

The time to access the data itself appears not surprisingly as the most expensive part, and the impact of the access to the different metadata with access control remains very limited (16% of the time).

Regarding the openness of our architecture, one can add new functionalities with minimal efforts. Indeed, new tools may be added with specific tool drivers in the architecture. New usage of metadata, ranging from access optimization, traceability, or any other taking a benefit from the semantics of the data will thus be easily addressed in the future.

VI. Related Works

The works on metadata management are very numerous, especially in the database community and are used for instance in mediators to aggregate schemas from heterogeneous databases. In distributed systems, and more specifically in High Performance architecture and more recently in Grid Computing, some researchers addressed the metadata management problem. In 1999, Choudhary et al. [4] provides a Meta Data Management System (MDMS) where system metadata are used to distribute and retrieve the data in a distributed system (according to its use in the application). MDMS used a single MDMS provider (thus difficult to scale well in Grid environments). Hybrid queries were not addressed nor dynamic metadata or extension like security and caching.

The Metadata Catalog (MCAT) designed in the SDSC Storage Resource Broker (SRB) [2] is very similar to our system. It manages descriptive and system metadata, and the SRB uses these to control and optimize the access to data. Differences with our approach are (1) the ability of a collection of DM^2s to communicate in a distributed way (2) the management of hybrid queries involving grid-related computation, and thus the dynamic enrichment of metadata and (3) our role based access control is more complete than in SRB.

More recently, Singh et al. [15] proposed a related work to MCAT, the MCS (Metadata Catalog Service) designed to handle logical metadata attributes. An application program contacts a local MCS client, this MCS client opens a communication with a MCS server where the metadata are retrieved from a MySQL database. Only basic queries are mentioned in the article (no hybrid queries). Security relies on the Grid Security Infrastructure (and in the future with the Community Authorization Service [12]) that does not suit all our access control requirements (see [14]). The study of the scaling of the architecture does not address the connections of numerous clients on the unique MCS server. The authors describe a general service for logical metadata access. We argue that our architecture has been designed in a general way and that it opens parallel and complementary researches for interleaving access-to and computation-on data and metadata, when the dynamic of both have to be handled.

Finally, the RepMec [11] catalog developed in the European DataGrid project, using Spitfire [3], is very similar to this latter, but is used mainly for file localization in the Grid.

VII. Conclusion

We presented in this article an extensible architecture that allows for secure and efficient access to medical data through the management of metadata. In our medical application, we have identified static and dynamic metadata, and we have exhibited the adequation of our platform for their handling, in terms of performance and scalability. We try here to extract some general methodologies to improve the management of sensisitive and dynamic metadata. We contribute through the Metadata Management group at the Global Grid Forum to help to propose an architecture for metadata management in OGSA [9].

References

[1] N. E. M. Association. *Digital Imaging and Communications in Medecine (DICOM)*. Rosslyn, Virginia, 2001. DICOM 3.

[2] C. Baru, R. Moore, A. Rajasekar, and M. Wan. The SDSC storage resource broker. In *Proceedings of CASCON'98*, Toronto, Canada, 1998.

[3] W. Bell, D. Bosio, W. Hoschek, P. Kunszt, G. McCance, and M. Silander. Project spitfire - towards grid web service databases, 2002.

[4] A. Choudhary, M. Kandemir, H. Nagesh, J. No, X. Shen, V. Taylor, S. More, and R. Thakur. Data management for large-scale scientific computations in high performance distributed systems. In *Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing*, pages 263–272, Redondo Beach, CA, USA, 1999. IEEE Computer Society Press.

[5] H. Duque, J. Montagnat, J.-M. Pierson, I. Magnin, and L. Brunie. DM^2: A Distributed Medical Data Manager for Grids. In *Biogrid 03, Tokyo May 12th to 15th 2003, proceedings of the IEEE CCGrid03*, 2003.

[6] European Data Grid. The datagrid project. http://eu-datagrid.web.cern.ch/eu-datagrid/, 2001.

[7] D. Ferraiolo and D. Kuhn. Role based access control. In *15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[8] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.

[9] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, 2002.

[10] A. S. Grimshaw, W. A. Wulf, J. C. French, A. C. Weaver, and P. F. Reynolds Jr. Legion: The next logical step toward a nationwide virtual computer. Technical Report CS-94-21, University of Virginia, 8, 1994.

[11] L. Guy, P. Kunszt, E. Laure, H. Stockinger, and K. Stokincker. Replica management in data grids. Technical report, Global Grid Forum Informational Document, GGF5, Edinbourg, Scotland, July 2002.

[12] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A community authorization service for group collaboration. In *Proceedings of the 2002 IEEE Workshop on Policies for Distributed Systems and Networks*, 2002.

[13] L. Seitz, J. Pierson, and L. Brunie. Key management for encrypted data storage in distributed systems. In *Proceedings of the second Security In Storage Workshop (SISW)*, 2003.

[14] L. Seitz, J. Pierson, and L. Brunie. Semantic access control for medical applications in grid environments. In *Euro-Par 2003 Parallel Processing*, volume LNCS 2790, pages 374–383. Springer, 2003.

[15] G. Singh, S. Bharathi, A. Chervenak, E. Deelman, C. Kesselman, M. Manohar, and S. Patil. A metadata catalog service for data intensive applications. In *Proceedings of the ACM/IEEE SuperComputing 2003 Conference*, pages 33–49, Phoenix, AZ, USA, 2003. IEEE Computer Society Press.