DM2: A Distributed Medical Data Manager for Grids

Hector Duque, Johan Montagnat, Jean-Marc Pierson, Lionel Brunie, Isabelle Magnin

DM^2: A Distributed Medical Data Manager for Grids*

H. Duque[ab], J. Montagnat[a], J.-M. Pierson[b], L. Brunie[b], and I.E. Magnin[a]
[a] CREATIS, CNRS UMR 5515, http://www.creatis.insa-lyon.fr/
[b] LISI, INSA de Lyon EA 629, http://lisi.insa-lyon.fr/
INSA, Bât. B. Pascal, 7 av. J. Capelle, 69621 Villeurbanne Cedex, France

Abstract

Medical data represent tremendous amount of data for which automatic analysis is increasingly needed. Grids are very promising to face today challenging health issues such as epidemiological studies through large image data sets. However, the sensitive nature of medical data makes it difficult to widely distribute medical applications over computational grids. In this paper, we review fundamental medical data manipulation requirements and then propose a distributed data management architecture that addresses the medical data security and high performance constraints. A prototype is currently being developed inside our laboratories to demonstrate the architecture capability to face realistic distributed medical data manipulation situations.

1 Medical applications on computational grids

Recently, computational grids [7, 8] achieved a large success among the distributed computing community. Many technical developments aim at providing a middleware layer allowing to submit remote jobs [2, 5, 9], store data [4], and get information on a distributed system [15]. But most importantly, from the user point of view, grids should allow transparent access to distributed resources and enable easy data and algorithms sharing.

Medical data analysis is a domain where grid technologies are very promising. Health centers are using an increasing number of digital 3D sensors for medical data acquisition [1], representing tremendous amount of data for which automatic analysis is needed. Grid technologies offer: (i) an increased computing power for complex modeling and simulation algorithms, (ii) a distributed platform with shared resources for different medical partners with similar data processing needs, (iii) a common architecture to access heterogeneous data and algorithms for processing, and (iv) the ability to process very large amounts of data (*e.g.* for epidemiological studies).

However, to remain transparent from the user point of view, a middleware layer should take into account the particular needs of medical applications. Although weakly structured, medical data have a strong semantic and metadata are very important to describe data content (such as images). Furthermore, medical data are sensitive and should only be accessible by accredited users, which makes data manipulation over a wide area network difficult.

In this paper, we address the medical data management related issues for grid computing. We first detail in section 2 medical data requirements. In section 3, we propose a novel data management architecture that is currently being investigated in our laboratories.

2 Medical data and security issues

2.1 Medical data

Although there is no universal standard for storing medical images, the most established industrial standard is DICOM (Digital Image and COmmunication in Medicine) [6]. DICOM describes both an image format and a client/server protocol to store and retrieve images on/from a medical image server. Most recent medical imagers implement the DICOM protocol. They play the role of acquisition device and image server communicating with their clients over a TCP/IP hospital network. The DICOM file format is made of a header containing metadata followed by one or several image slice(s) in a single file.

The metadata contains two kinds of information:

*This work is partly supported by the IST European DataGrid project (http://www.edg.org/), the French ministry for research ACI-GRID project (http://www-sop.inria.fr/aci/grid/public/) and ECOS-Nord Committee (action C03S02).

- sensitive **patient-related** metadata such as the patient name, sex, age, the radiologist name, the hospital, etc.
- **image-related** metadata such as the image acquisition device, constructor, and parameters, the acquisition date, the number of images stored, the size of images, the field of view, etc.

To the patient information can be added all clinical metadata that are not originally part of the image acquisition such as notes by medical experts. Although patient metadata is the most sensitive part of the data, no medical data, including the image content, should ever be accessible to unauthorized users.

Today, medical data are often stored and archived inside each medical image producer (hospitals, clinics, radiology centers...). The medical record (image files and metadata) of one patient is distributed over the different medical centers that have been involved in his healthcare. Medical data are usually disconnected from the outside world to solve security issues. Several Picture Archiving and Communication Systems (PACS) [10] have been developed to provide data management, visualization, and, to some extent, processing. However, they are restricted to data management inside each hospital and hardly address the problems arising when considering a global system with communication of sensitive data between sites.

2.2 Requirements for medical data on the Grid

Data management and replication mechanisms [14] proposed by grid middlewares mainly deal with flat files. Data access control is handled at a file level. In the DataGrid project for instance, user authentication relies on the asymmetric key-based Globus Grid Security Infrastructure layer (GSI) [3]. This infrastructure does not take into consideration metadata and does not address patient record distribution. Therefore, we investigate the creation of a *Distributed Medical Data Manager* unit (abbreviated as DM^2 later in this document) that interfaces with the grid middleware. It should provide:

- Reliable and scalable storage for images and metadata produced by medical imagers. This includes connection to the grid file replication mechanism and a metadata location service granting access to distributed medical records (see section 3.2.2 for details). To face scalability and reliability issues in a wide area environment, replication of metadata also appears necessary.
- Data and associated metadata should be synchronized by the information system as they are semantically connected (they should have the same lifetime, same access patterns...).
- Through a proper query service, metadata should describe a dataset and allow it to be processed. This implies a proper interface between the grid job submission service and the data/metadata manager.
- Encryption is needed to restrict access to authorized users.
- A fine user identification mechanism should control access rights to the data.
- Finally, medical data traceability is an important feature: a medical doctor is often interested in knowing from where a computed data originates, and conversely, it may be useful for optimization reasons to remember which computations have already been carried out a given data set and what results were produced.

Processing or querying data over a grid raise problems of confidentiality and confidence that the user may have in the grid security infrastructure. Ideally, medical data should not be accessible by any unaccredited user, including system administrators of sites where the data are transported for computation. To ensure a reasonable confidentiality level, we plan to decouple the sensitive metadata from the image data. The metadata will only be stored on a limited number of trusted sites, where administrators are accredited, and can only be sent to accredited user stations. The image data can be stored, replicated, and manipulated on standard grid nodes given that it is encrypted to make its content retrieval difficult.

3 Managing distributed medical data

3.1 A sample usecase

To illustrate our proposal for a DM^2, we will consider the following usecase. A cardiologist looks for heart images similar to a case of one of his patients to confirm his diagnosis. He want to rank existing images through a similarity score resulting of a computation involving his patient image and an image database. Once the images are ranked, he need to visualize the most similar cases and their attached diagnoses. The diagnosis he makes for his patient can be recorded in the information system for improving the existing database. In technical terms, the cardiologist needs to:

- Access a large data set made of comparable heart images that are distributed over different hospitals.

- Make computations (similarity measurements) on a large number of images in a very limited time.

The grid can help on both aspects by providing a very large distributed image database and the computing power needed. However, this implies that there is a full and secured interface between the grid computation services, the medical data servers, the associated metadata, and the user interface.

3.2 Key issues in designing the DM^2

A distributed system is made of several interconnected computers and a shared state describing the cooperation of these computers [11]. To respond the requirements described in section 2.2 the DM^2 is designed as a complex system involving multiple grid service interfaces and several interacting processes geographically distributed over an heterogeneous environment. It is a gate to the grid as well as an intermediary (proxy) between the grid and a set of trusted medical sites. Its complexity has motivated us to first propose an architecture describing the DM^2 components and second to implement our system as one possible instance of this architecture. To tackle the DM^2 complexity, we propose the multi-layers architecture outlined in section 3.2.1. We also need to interface the DM^2 with underlying grid services as detailed in section 3.2.2.

3.2.1 DM^2 layered architecture

The DM^2 needs to interconnect with existing services on the internals of which we have no control. We will refer to *engines* to designate the DM^2 services that we develop to avoid confusions. Each engine is composed of a set of independent processes which interact by exchanging messages. We design each *Distributed System Engine* (DSE) through a layered architecture that takes into account the requirements for high performance and data integrity. The architecture increases in semantic significance through five layers (see figure 1). At the lowest level, DSE^0, the system is made of processes that can communicate through a message passing kernel. The DSE^1 level brings atomic operations (*transactions*) to process complex requests composed of many messages. The upper layer DSE^2 deals with distribution over several engines. On top of the distribution layer come the application (DSE^3) and user interface (DSE^4).

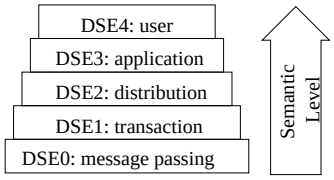

Figure 1. DSE layers

3.2.2 Interfacing medical data servers with the grid middleware

The European DataGrid (EDG) middleware proposes a standard storage interface to the underlying mass storage systems. Through this interface, the middleware may access files on distributed and heterogeneous storage pools. Grid-enabled files are handled by a *Replica Manager* (RM): to ensure fault tolerance and provide a high data accessibility service, files are registered into the RM and may be replicated by the middleware in several identical instances. The first file registered into the RM is a *master file*. Other instances are *replicas*. When a file is needed, the grid middleware will automatically choose which replica should be used for optimizing performances. Having multiple instances of a file also increase its availability since connection errors are likely to happen in a wide scale distributed system. To solve coherency problems, replicas are accessible in read only mode and modifying a master file invalidates all its replicas. To ease files manipulation, grid wide *Logical File Names* (LFN) are used to identify each logical data (*i.e.* a master and all its replicas).

In the hospital, each image can be made up of one or several DICOM files representing portions of the imaged body. The DM^2 plays a double role to interface DICOM servers with the grid middleware as illustrated in figure 2:

- It makes an interface between the grid RM and the DICOM servers.
- It provides a standard storage interface allowing to exchange files with other grid services.

For each new DICOM image or set of DICOM images (depending on the semantic of the DICOM series) produced by an imager, a LFN is created and registered into the RM. The DICOM files thus becomes, from the grid side, a master file. There is not necessarily a physical file instance behind this LFN but rather a virtual file made up of a set of DICOM files, that can be reconstructed on the fly by the DM^2 if a request for this LFN comes in. For efficiency reasons, assembled files are cached on a scratch space before being sent outside. The DM^2 also stores metadata and establishes a

link between an LFN and its patient- or image-related metadata.

The DM^2 storage interface ensures data security by anonymizing and encrypting on the fly images that are sent to the grid. Replicas of a medical image may exist on any grid storage node, given that encryption forbid data access without encryption keys. These keys are stored with the patient-related metadata on trusted sites only. In order to ensure data integrity, the grid storage interface does not allow the master files stored on the DICOM server to be deleted.

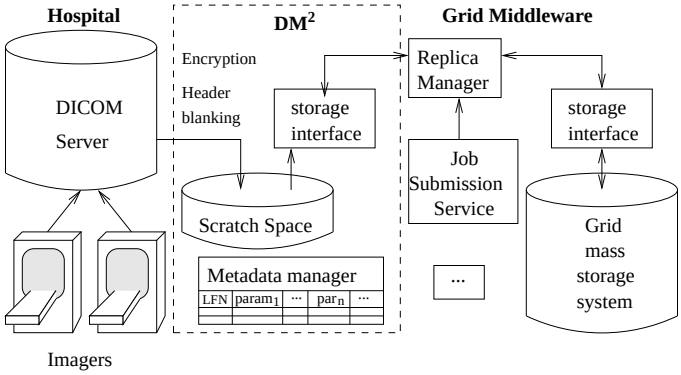

Figure 2. DM^2 interface between the medical imagers and the grid

3.3 DM^2 layers

As mentioned earlier, we decomposed the DM^2 architecture in 5 layers to tackle its complexity. This section further describes each layer's role.

3.3.1 DSE^0: message passing

The core of the DSE^0 is a message passing kernel in charge of the efficient transmission of messages between the DM^2 processes. Our kernel implementation is based on Inter-Process Communication (IPC) services. The DSE^0 transmits messages between the IPC kernel and the external network for access to both local and remote services.

3.3.2 DSE^1: transactions

On top of the simple message transportation layer, DSE^1 implements atomic operations (transactions) made of multiple sub-operations. A transaction succeeds if and only if all sub-operations succeed. If a failure occurs the system is left in a coherent state. It offers the ACID properties: Atomicity, Consistency, Isolation and Durability [11].

A transaction involves multiple *requests* to external services and internal engines. To deal efficiently with complex transactions, we introduce the notion of *tasks*. Tasks are sets of sequential requests and transactions are sets of sequential and/or concurrent tasks (to shorten the processing time). We implemented transactions, tasks, and requests through four kinds of specialized processes that we will refer to as *drivers*:

- The TRransactions Drivers (TRD) are processes which can manage a whole transaction made up of a set of sequential or parallel tasks. A transaction could imply having access to different external services (tasks).
- The TasKs Drivers (TKD) are processes in charge of doing a specialized part of a transaction, which could imply getting sequential access to an external service or to a set of request drivers.
- The ReQuest Drivers (RQD) are in charge of accessing the remote components such as other engines and external servers. They solve low level issues such as connection management. These drivers transmit messages and receive responses that they route to the calling processes.
- The TOol Drivers (TOD) are processes which perform internal operations for debugging support or improving the efficiency. Examples of such processes are the caching of requests and results, logging, security checking, and tracing operations.

A DM^2 engine works as follows: (i) A message arrives at a transaction driver and a transaction is initiated. (ii) The transaction starts different concurrent tasks, using independent processes (TKD). (iii) Each task get access to the requests drivers (RQD) so that it can reach the external services. (iv) The request drivers (RQD) opens connections and send messages to the external services. (v) Each driver uses the tools it needs (TOD). See section 3.4.2 for a concrete example.

3.3.3 Upper layers

DSE^2 brings additional distributed facilities on top of the two lowest levels. It is in charge of localizing data and services, transmitting request to proper hosts, etc. It may take advantage of completely decentralized Peer-to-Peer (P2P) techniques or to semi-hierarchical tree structures such as LDAP for distributed data localization.

DSE^3 is the application layer, offering a programming interface (API) so that an application can be built on top of the underlying distributed system.

DSE[4] is the user layer. It offers high level access to data, metadata and algorithms registered in the system. It can be implemented as web portals or graphical applications. For instance, a user might have access to similarity algorithms published by the DM[2] architecture (through this portal) that he wants to apply to a subset of images (according to its access rights).

3.3.4 Extensibility and Interfaces

The architecture allows external applications to interact with the distributed system, and to execute locally on the same node or remotely. The system can access other servers offering additional services (*e.g.* grid services) or be accessed by clients trying to take advantage of the DM[2] services. In this way additional functions such as cache, security, file transfer and encryption, database access, etc, can be easily interfaced, and designed as independent modules for easing software development.

3.4 Using the DM[2] in real applications

A prototype is currently under development to demonstrate the relevance of the proposed architecture in realistic medical applications. At the moment, we have implemented the interface to a DICOM server (*Central Test Node*), metadata handling through SQL tables with a secured access interface, but we do not have a fully secured and distributed system yet. Our prototype is therefore an assembly of DSE[0] and DSE[1] processes (message passing and basic transactions) following the above architecture.

3.4.1 DM[2] software components

Let us consider the usecase described in section 3.1. In order to set up this application we implemented:

- A set of request drivers for issuing requests to and getting results from the grid services. The *DICOM RQD* accesses the DICOM server where medical images are stored and the *metadata RQD* is a driver for the database service where image metadata are stored.
- Each one of these services has an associated multiprocess task driver which is able to execute concurrent demands. The *DICOM TKD* can make parallel transfer of DICOM files for instance.
- In addition, a *communication daemon* TRD has been implemented in order to receive messages from the network side and to start the execution of transactions.

3.4.2 Detailed usecase

First, the cardiologist enters a query (*e.g.* find the MRI of Mr X acquired yesterday in this hospital) through the DM[2] user interface. The DM[2] sends a query for retrieving metadata to the grid metadata interface through the *metadata RQD* and *TKD*. The user authorizations to access the data are checked by the external metadata service and the patient file logical identifier and its associated parameters (imaging modality, region of interest, dynamic sequence, MR acquisition parameters, etc) are returned to the user interface.

A request is made to find all images comparable to the image of interest (same body region, same acquisition modality...) and for which a medical diagnosis is known. The DM[2] layer 2 should be used to distribute the request on all hospitals with metadata services. In the current implementation, the single metadata service is queried through the *metadata RQD* and *TKD* again. The logical identifier of all images matching the patient source file parameters are returned.

A request is then made for computation of a similarity measure [12, 13] between the patient image and each image resulting from the previous query. The job submission service of the grid middleware is used to distribute computations over available working nodes. For each job started, the grid replica manager triggers a replication of the input files to process onto grid computation nodes. When requesting files to the DM[2] storage interface, the DM[2] queries the DICOM server, assembles MR images on the fly onto its scratch space and returns images to grid nodes. Figure 3 details this operation: on top, the grid middleware triggers a DM[2] transaction for getting an image. (1) It first makes security checks. (2) It then accesses the database (*metadata TKD*) to locate the DICOM files. (3) A *cache TOD* (when available), can be used to improve the latency transfer. (4) Assuming the cache does not contain the requested file, it should be copied from the DICOM server. The DM[2] queries 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) Finally, the image is stored into the cache and returned to the grid calling service.

4 Conclusions

Medical image processing over the grid opens new opportunities for applications involving large medical datasets. However, full deployment of medical applications require medical sites to be interconnected and grid middlewares to provide secure and efficient access

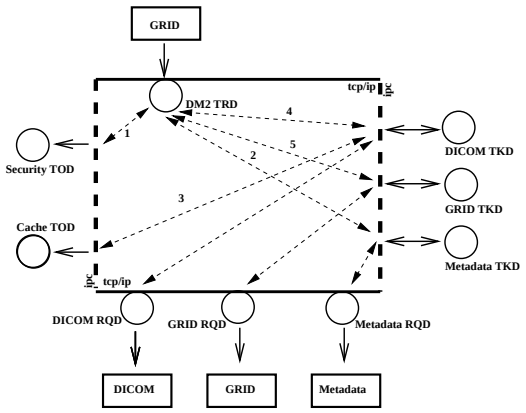

Figure 3. DSE[1] usage example: retrieving a medical image from the DICOM server.

to the distributed and sensitive medical data. The semantic content of medical data should also be taken into account by developing grid-wide tools to manage associated metadata.

The architecture proposed in this paper allows us to build a complex distributed system, taking advantage of classical theory (transactions concept) and proposing solutions to implement a high performance data manager (decomposition of transactions in concurrent tasks and requests). The DM^2 system allows the physicians to get secure access to their patients' images and to send hybrid requests over huge databases.

We implemented a first prototype to demonstrate the relevance of the DM^2 for realistic medical applications. On-going work concerns data security and distribution. Many other aspects related to medical data management could not be addressed in this short paper including the need for tracking data origin and logging data processing, optimization procedures such as data and request caching.

References

[1] R. Acharya, R. Wasserman, J. Sevens, and C. Hinojosa. Biomedical Imaging Modalities: a Tutorial. *Computerized Medical Imaging and Graphics*, 19(1):3–25, 1995.

[2] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao. Application-level Scheduling on Distributed Heterogeneous Networks. In *Supercomputing*, Pittsburgh, PA, USA, November 1996.

[3] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and Welch V. A National-Scale Authentication Infrastructure. *IEEE Computer*, 33(12):60–66, 2000.

[4] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23(3):187–200, July 2000.

[5] Karl Czajkowski, Ian Foster, Nick Karonis, Carl Kesselman, Stuart Martin, Warren Smith, and Steven Tuecke. A resource management architecture for metacomputing systems. *Lecture Notes in Computer Science*, 1459:62, 1998.

[6] DICOM: Digital Imaging and COmmunications in Medicine. http://medical.nema.org/.

[7] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, 15(3), 2001.

[8] Ian Foster and Carl Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, July 1998.

[9] Francesco Giacomini, Francesco Prelz, Massimo Sgaravatto, Igor Terekhov, Gabriele Garzoglio, and Todd Tannenbaum. Planning on the grid: A status report. ppdg-20, particle physics data grid collaboration., October 2002.

[10] H. K. Huang. *PACS: Picture Archiving and Communication Systems in Biomedical Imaging*. Hardcover, 1996.

[11] Sape Mullander, editor. *Distributed Systems*. Addison Wesley, 1993.

[12] G.P. Penney, J. Weese, J.A. Little, P. Desmedt, D.LG. Hill, and D.J. Hawkes. A Comparison of Similarity Measures for Use in 2D-3D Medical Image Registration. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI'98)*, volume 1496 of *LNCS*, pages 1153–1161, Cambridge, USA, October 1998. Springer.

[13] A. Roche, G. Malandain, X. Pennec, and N. Ayache. The Correlation Ratio as a New Similarity Measure for Multimodal Image Registration. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI'98)*, volume 1496 of *LNCS*, pages 1115–1124, Cambridge, USA, October 1998. Springer.

[14] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, and B. Tierney. File and object replication in data grids. In *10th IEEE Symposium on High Performance and Distributed Computing (HPDC2001)*, August 2001.

[15] B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolsky, and M. Swany. A grid monitoring architecture. tech. rep. gwd-perf-16-2, global grid forum, January 2002.