Safe Distributed Architecture for Image-based Computer Assisted Diagnosis

Sébastien Varrette*† and Jean-Louis Roch*
*MOAIS Team (INRIA-CNRS-UJF-INPG)
LIG-IMAG Laboratory
Montbonnot Saint Martin, FRANCE

Johan Montagnat
CNRS, I3S lab, RAINBOW team
Sophia Antipolis, FRANCE

Jean-Marc Pierson
LIRIS Laboratory
Lyon, FRANCE

Ludwig Seitz
SPOT Laboratory
Kista, SWEDEN

Franck Leprévost†
†LACS Laboratory
Luxembourg, LUXEMBOURG

***Abstract*— Existing electronic healthcare systems based on PACS and Hospital IS are designed for clinical practice. Yet, both for security, technical and legacy reasons, they are often weakly connected to computing infrastructures and data networks. In the context of the RAGTIME project, grid infrastructures are studied to propose a cheap and reliable infrastructure enabling computerized medical applications. This raises various concerns, in particular in terms of security and data privacy. This paper presents the results of this study and proposes a complete grid-based architecture able to process medical image for assisted diagnosis in a secured way. Using this infrastructure, care practitioner are able to execute the application from any machine connected to the Internet, therefore improving their mobility. Medical image analysis jobs are certified to be correct using the latest advances in result checking and fault-tolerant algorithms provided in [1], [2]. The architecture has been successfully deployed and validated on the Grid5000 large scale infrastructure.**

I. Introduction

The RAGTIME project[1] federates researchers in the grid computing community around a common goal: the management of medical information. For that purpose, grids and more particularly grids of clusters [3] are studied to provide a cheap distributed computing infrastructure complementary to clinical PACS[2] for medical application. The project aims to demonstrate how grid technologies can improve the cooperation between PACS located in distant hospitals and enable medical image analysis procedures.

A grid of cluster corresponds to a cluster aggregation through the Internet with a remote access for users. This topology is particularly adapted to represent the network that would interconnect the PACS.

The experiment described in this article has the ambition to convince care practitioner of the improvement and the flexibility provided by such an architecture (the latter point is generally seen as incompatible with this technology for end-users). For that purpose, this paper illustrates an application of breast cancer lesions detection in mammograms using statistical comparison on a database of studied cases (see figure 1). In practice, the database should be located on PACS that are seen as a secure distributed storage grid for this application. On the other side, a computing grid composed of interconnected clusters (either located in hospitals or in supporting institutions) executes comparison algorithms to evaluate the similarity between a new mammogram submitted by a doctor to those registered in the storage grid.

[1]Literally "Rhône-Alpes : Grille pour le Traitement d'Informations Médicales" `http://liris.univ-lyon2.fr/~miguet/ragtime/`

[2]Picture Archiving and Communication Systems

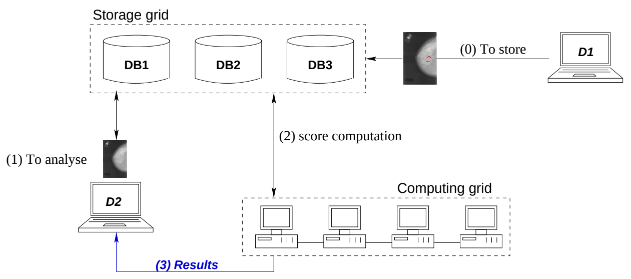


Fig. 1. Application for mammograms comparison

Of course, this application raises various constraints, mainly in terms of security and privacy:

- The system should be accessible from any computer connected to the Internet.
- Only authorized users (typically a doctor) should be allowed to use this application and access the resources (machines or data) required.
- Communications during the process between the resources should be encrypted to guarantee their privacy and integrity.
- Medical images sent on the computing grid have to be anonymized to guarantee patient privacy even in case of a resource corruption.
- Data on the storage grid should be securely stored.
- The system should remain operative even in case of resources corruption.
- The jobs should be cleverly scheduled on the grid

All these constraints are addressed by the proposed architecture. Point 4 should normally be addressed by the PACS. Yet, for obvious security reasons, access to a real PACS in production mode has not been possible. In this article, the storage grid is achieved by a simple database, even if we negotiate access to the distributed database on EGEE[3]. We therefore detail in §II-D how data are securely stored in this model. It is important to notice that as soon as unsafe resources are used (as in pervasive architecture), it is impossible to completely trust the results computed [4]. That's why some algorithms are considered in §II-B to certify the computed results. The remaining sections are organized as follows: §II expounds and justifies the components of the proposed architecture. §III details the protocol used while §IV concludes this article and explain future works we plan to add to improve this experiment.

II. Architectural Components

A. Authentication System for Grid Access

Designing a robust authentication system in distributed environments has been extensively studied [3], [5], [6]. Efficient solutions depend on the grid topology. As for grids of clusters, the authors of [3] demonstrate an adapted and efficient solution based on LDAP servers that broadcast authentication information. In terms of security, LDAP provides various guarantees thanks to the integration of cypher and authentication standard mechanisms (SSL/TLS, SASL) coupled with Access Control Lists. All these mechanisms enable an efficient protection of transactions and access to the data incorporated in the LDAP directory. The proposed solution is currently used as the authentication system of Grid5000[4]. It enables access to the Grid5000 grid - and for the context of this paper to the execution platform - from any computer connected to the Internet. Yet, LDAP could easily use other authentication technologies such as smartcards - this kind of authentication will be investigated in future works.

B. Ensuring Computation Resilience

Resilience in grid execution is a prerequisite that should be embedded in the application: at this scale, component failures, disconnections or results modifications are part of operations, and applications have to deal directly with repeated failures during program runs. This integration can be done in a cross-platform way using a portable representation of the distributed execution: a bipartite Direct Acyclic Graph $G = (\mathcal{V}, \mathcal{E})$. the first class of vertices is associated to the tasks (in the sequential scheduling sense) whereas the second one represents the parameters of the tasks (either inputs or outputs according to the direction of the edge). Such a graph is illustrated in figure 2.

Using this representation, the authors in [7], [2] propose portable fault-tolerance mechanisms for heterogeneous multi-threaded applications. The flexibility of macro dataflow graphs

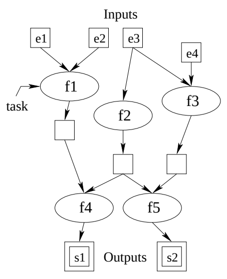


Fig. 2. Instance of a data-flow graph associated to the execution of five tasks $\{f_1, ..., f_5\}$. The input parameters of the program are $\{e_1, ..., e_4\}$ whereas the outputs (i.e the results of the computation) are $\{s_1, s_2\}$

has been exploited to allow for a platform-independent description of the application state. This description resulted in flexible and portable recovery strategies with a low overhead that only required the existence of a checkpoint server deployed on a set of safe resources. This server stores the dataflow graph of the execution provided by the Kernel for Adaptive, Asynchronous Parallel Interface (KAAPI). KAAPI is a C++ library that allows to program and execute multi-threaded computations with dataflow synchronization between threads. The same approach can be exploited in this paper to ensure resilience to crash fault of computing resources.

Alternatively, any error on the analysis of an image computed on a grid could have dramatic consequences on the resulting diagnosis. We do the "optimistic" assumption that even if the majority of resources will compute correctly, they cannot be fully trusted. It is therefore important to reassure the care practitioner that the computed results are correct and have not been tampered by a corrupted resource. This requires efficient error checking algorithms able to certify the correctness of the computation. In this area, dataflow graphs are also used [1], [8] and provide a tunable probabilistic certification. These research also assumed the availability of safe resources gathering a checkpoint server (eventually distributed) together with a controller (or verifier) used to safely re-execute some tasks of the program.

Both mechanisms – either for fault-tolerance or error checking – have to be used in the target medical application. This leads to the infrastructure presented in figure 3 in which the resources have been divided in two classes:

1) A limited number of safe resources host the checkpoint server and the verifiers. As it will be seen in §II-E, the farm daemon of the μgrid middleware will also be hosted on these resources.
2) the other resources, mentioned as "unsafe", constitute the real computing grid and are divided among the different hospitals and involved institutions.

C. DICOM Image anonymization

In this experiment, medical images are encoded using the standard DICOM format (Digital Image and COmmunication in Medicine). DICOM files contain both image data and metadata headers containing sensitive patient identifying information such as name, sex, data acquisition site, etc. Before

[3]Enabling Grids for E-sciencE `http://public.eu-egee.org/`

[4]The Grid5000 project aims at building a highly reconfigurable, controlable and monitorable experimental Grid platform gathering 9 sites geographically distributed in France featuring a total of 5000 CPUs - `https://www.grid5000.fr`

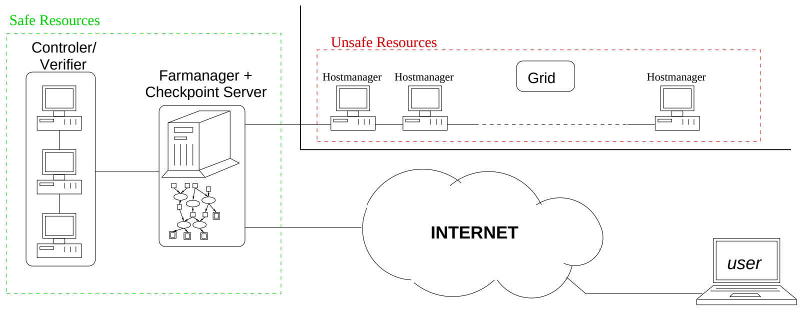


Fig. 3. Resources hierarchy and mandatory components for portable fault-tolerance and error-checking algorithms

sending any data to the computing grid infrastructure, the DICOM images have to be anonymized to ensure patient privacy. This is simply done by whipping all metadata out of the DICOM files.

D. Secured Storage and Access

Since the image files and the associated meta-data in the DICOM image format must be considered sensitive information, one goal is to protect them against unauthorized disclosure while they are on the storage Grid. Archived images cannot be fully anonymized, since we need to keep the person related meta-data that are very important in many medical diagnosis procedures. A convenient solution is to encrypt all sensitive data before it is stored onto the Grid.

When using data encryption, the problem arises of how to make it possible for authorized users to decrypt the data in order to gain access to its contents. Therefore we need a mechanism that makes decryption keys accessible for authorized users, while not compromising the security of the data encryption. Furthermore access to the keys needs to evolve dynamically with the individual access rights of the users, without requiring external intervention of an administrator. Finally we want to have some degree of safety in the key storage, so that the loss of one or more keys do not cause the loss of data it encrypts.

In order to make the keys available we store them on key servers that are not necessarily part of the Grid itself. These key servers use an access control mechanism to determine who may access which decryption key. The access control mechanism used by the key servers should mirror exactly the file access permissions of the users on the Grid. We have implemented an access control system called *Sygn* that can be used to achieve this. The use of Sygn in the RAGTIME environment is described in a former paper [9].

In order to make the key server more resilient to attacks and breakdowns, we do not store entire keys on a single key server. Instead we use several key servers and split up the keys we want to store into key-shares, using Shamir's secret sharing algorithm [10]. This gives us two considerable advantages: first, a successful attack on one key server does not expose actual keys. Attackers will need to successfully attack a number of key servers equal to the chosen threshold value of the secret sharing algorithm in order to be able to reconstruct the actual keys. Second, the algorithm allows for the creation of redundant key shares, meaning that you only need *any n out of m* (with $n < m$) key-shares to reconstruct a key. We therefore gain some redundancy if a key server should be unavailable or even loose its key related data.

The actual data encryption, creation and storage of key-shares on key servers is performed by a local tool on the machine of the user that produces the image. We have implemented a prototype of this encrypted storage architecture called *CryptStore* which is described in more detail in our publication *Encrypted Storage of Medical Data on a Grid* [11].

E. μgrid

Grid5000 is an experimental platform for grid computing research that is not making any assumption on the middleware to be used. Instead, Grid5000 users are deploying the middleware they need for their research and experiments. We have deployed the μgrid middleware [12] over the Grid5000 infrastructure. μgrid is a lightweight middleware prototype that was developed for research purposes and already used to deploy applications to medical image processing [13].

The μgrid middleware was designed to use clusters of PCs available in laboratories or hospitals. It is intended to remain easy to install, use and maintain. Therefore, it does not make any assumption on the network and the operating system except that independent hosts with private CPU, memory, and disk resources are connected through an IP network and can exchange messages via communication ports. This matches the Grid5000 platform. The middleware provides the basic functionalities needed for batch-oriented applications: It enables transparent access to data for jobs executed from a user interface. The code of μgrid is licensed under the GNU Public License and is freely available from the authors web page.

In the μgrid middleware a pool of hosts, providing storage space and computing power, is transparently managed by a *farm manager*. This manager collects the information about the controlled hosts and also serves as entry point to the grid. The μgrid middleware is packaged as three elements encompassing all services offered:

1) A host daemon running on each grid computing host that manages the local CPU, disk, and memory resources. It is implemented as a multiprocesses daemon forking a

new process for handling each task assigned. It offers the basic services for job execution, data storage and data retrieval on a farm.

2) A farm daemon, running on each cluster, that manages a pool of hosts. It is implemented as a multithreaded process performing lightweight tasks such as user commands assignment to computing hosts.
3) A user interface that provides communication facilities with farm daemons and access to the grid resources.

Although logically separated, the three μgrid components may be executed as different processes on a single host. The communications between these processes are performed using secured sockets. Therefore, a set of hosts interconnected via an IP network can also be used to run these elements separately.

The μgrid middleware offers the following services:

- User authentication through X509 certificates. Certificates are delivered by a certification authority that can be set up using the sample commands provided in the openSSL distribution.
- Data registration and replication. The middleware offers the virtual view of a single file system though data are actually distributed over multiple hosts. Files on the hosts need to be registered at the farm to be accessible from the grid middleware. Furthermore data can be transparently replicated by the middleware for efficiency reasons.
- Job execution. Computing tasks are executed on the grid hosts as independent processes. Each job is a call to a binary command possibly including command line arguments such as registered grid files.

These functionalities are handled by the μgrid components as follows. The farm daemon role is to control a computing farm composed of one or more hosts. It holds a database of host capacities, grid files, and a queue of scheduled jobs. MySQL is used as database back-end. When started, the farm daemon connects to the database back-end. If it cannot find the μgrid database, it considers that it is executed for the first time and sets up the database and creates empty tables. In the other case, it finds in the database, the list of grid files that have been registered during previous executions and the hosts where files are physically instantiated.

The host manager role is to manage the resources available on a host. When started, it collects data about the host CPU power, its available memory and the available disk space. It connects to the farm manager indicated on command line or in a configuration file to which it sends the host information. The host manager encompasses both a data storage/retrieval service and a job execution service.

To make use of the system, a user has to know a farm manager to which its requests can be directed. For convenience, the latest farm managers addressed are cached in the user home directory. Through the user interface, a user may require file creation, replication, or destruction, and jobs execution. These requests are sent to a farm manager which is responsible for locating the proper host able to handle the user request. To avoid unnecessary network load, the farm manager does not interfere any longer between the user interface and the target host, it only provides the user interface with the knowledge of the target host and then let the user interface establish a direct connection with the host for completion of the task. The system is fault tolerant in the sense that if the farm manager becomes unreachable (*e.g.* due to a network failure or the process being killed), the user interface parses the list of cached farm managers for completing the request. Similarly, if a worker node does not respond, it is declared ”down” and removed from the farm manager list of known living hosts until it restarts and registers again. The user interface consists of a C++ API. A single class enables the communication with the grid and the access to all implemented functionalities. A command line interface has also been implemented above this API that offers access to all the functionalities through four UNIX-like commands (ucp, urm, and uls for file management similarly to UNIX cp, rm, and ls commands respectively plus usubmit for starting programs execution on the grid).

F. *Analyzing medical images*

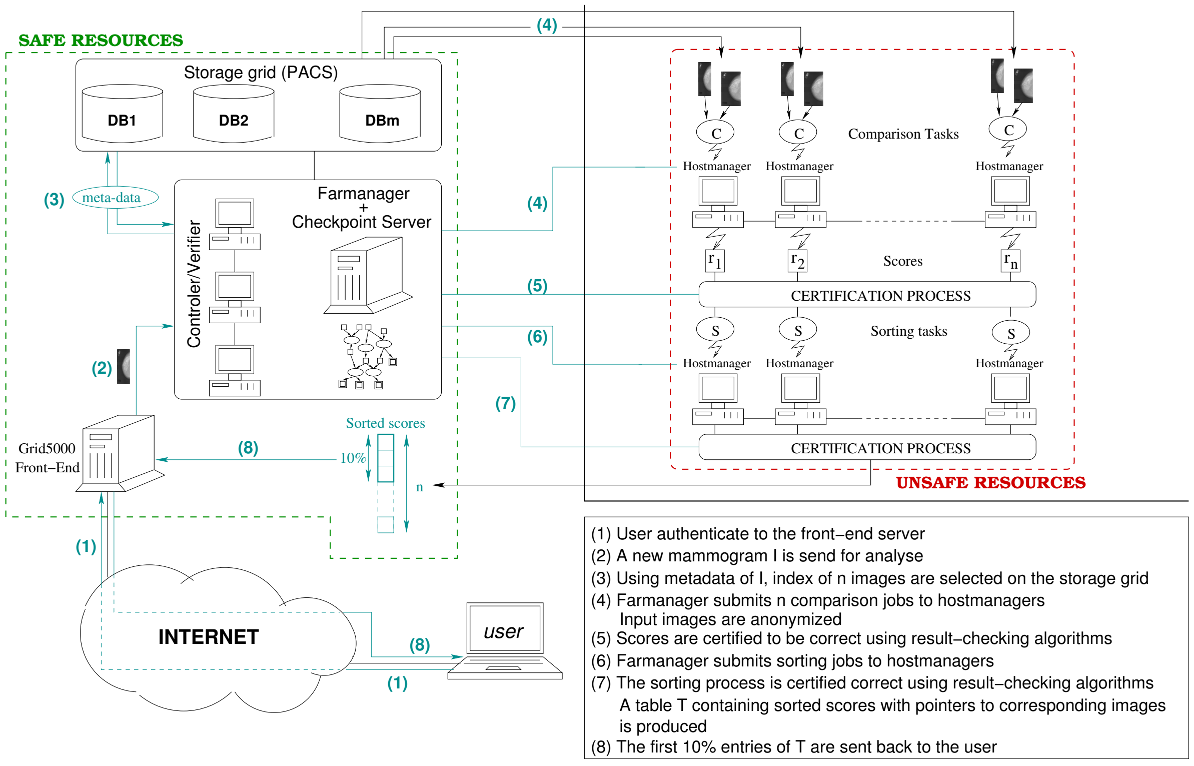
Many computerized medical image analysis algorithms are available today. Grid are particularly well suited to tackle very compute intensive applications like those requiring full image databases analysis. Indeed, massive data parallelism can often be exploited on such applications to distribute the workload over a grid infrastructure [14].

Computer Assisted Diagnosis techniques rely on target image comparison against annotated reference databases. The image comparison techniques greatly varies depending on the concrete medical objectives researched. Some global image indexing and analysis techniques have been proposed in the literature, both based on global image descriptors (histograms, global filter responses, etc) and local area features (local intensity and texture analysis, etc) [15]. Although some interesting usecases can be implemented, the consideration of a precise medical parameter to be extracted requires adaptation of these generic parameters.

In this paper, we are considering an application to breast cancer lesions detection in mammograms. The objective is to provide a computerized double reading of mammograms: A computer software selects images with an identified risk of malignant lesions for expert reading by a trained radiologist. Algorithms that may produce false positive (false alarms) but no false negatives (no malignant cases ignored) can be used for such an application. Image analysis procedures for mammograms based on a large number of local image descriptors have been proposed *e.g.* in [16].

G. *Sorting Algorithm*

Image analysis for assisted diagnosis first consists of comparison jobs which results have to be sorted. This can be done either on safe resources (the verifiers typically) or on the computing grid. The choice depends on the number of safe resources available. In the first case, $\mathcal{O}(n \log n)$ comparisons should be checkpointed to ensure the sorting of n scores. This approach should be prefered in general. In the later case, one



(1) User authenticate to the front−end server
(2) A new mammogram I is send for analyse
(3) Using metadata of I, index of n images are selected on the storage grid
(4) Farmanager submits n comparison jobs to hostmanagers
Input images are anonymized
(5) Scores are certified to be correct using result−checking algorithms
(6) Farmanager submits sorting jobs to hostmanagers
(7) The sorting process is certified correct using result−checking algorithms
A table T containing sorted scores with pointers to corresponding images is produced
(8) The first 10% entries of T are sent back to the user

Fig. 4. Detailed protocol for the RAGTIME Demonstration

should consider auto-tolerant algorithm to complement result-checking approach (see §II-B). This approach is required when safe resources are confined to limited embedded system and/or have a limited computing power. To facilitate the certification, we consider sorting algorithms composed of only one type of tasks. This leads us to sorting networks analyzed by Batcher [17] and Ajtai, Komlos, and Szemeredi [18]. Such a network consists of n registers and a collection of comparators, where n is the number of items to be sorted. Each register holds one of the items to be sorted, and each comparator is a 2-input, 2-output device that outputs the two input items in sorted order. The comparators are partitioned into levels so that each register is involved in at most one comparison in each level. The depth of the network is defined to be the number of levels in the network, and the size of the network is defined as the number of comparators in the network. Extending this model to our application, an algorithm only composed of comparator tasks has been considered. One can show that this algorithm requires at least $\Omega(n \log n)$ comparators and $\Omega(\log n)$ levels and this bound is reached using the AKS network [18]. As the best sequential algorithm has a time complexity of $\mathcal{O}(n \log n)$, the best improvement that can be expected using n processors is $\mathcal{O}(\log n)$ so that the AKS network is optimal except for a constant. In practice, the constant hidden under the $\mathcal{O}$ notation in AKS makes it less efficient than the bitonic sort of Batcher [17] (with size $\mathcal{O}(n \log^2 n)$ and depth $\mathcal{O}(\log^2 n)$) that should be prefered.

It remains to make this algorithm auto-tolerant to comparator tasks failures. The destructive fault model introduced in [19] has been considered: a faulty comparator task with inputs x and y can output $f(x, y)$ and $g(x, y)$ where f and g can be any of the following functions: x, y, $min(x, y)$ or $max(x, y)$. In the case of random faults, and given a n-item sorting network with depth d and size N, Assaf and Upfal showed how to construct a network with $\mathcal{O}(N \log N)$ comparators and $\mathcal{O}(d)$ levels that (with high probability) can sort n items even if a constant fraction of the comparators are faulty. Applied to the AKS network, this leads to size $\mathcal{O}(n \log^2 n)$ and Leighton & Ma in [20] demonstrates that this is an optimal size. With the bitonic sort algorithm, an algorithm with size $\mathcal{O}(n \log^3 n)$ (the checkpoint cost) and depth $\mathcal{O}(\log^2 n)$ is obtained.

III. Experimental Protocol

As mentioned in §II-B, the availability of safe resources is assumed. They will host the controllers, the checkpoint server and the farm manager. In addition, the storage grid, the front-end server and the key server are supposed on these resources. Concerning the image database, §II-D demonstrates how to provide a secure storage and access. The remaining resources compose the computing grid and are supposed unsafe. They each run a hostmanager daemon required by the μgrid middleware (see §II-E). The exact protocol of the experiment developed is summarized in the figure 4. It combines the architectural components detailed in §III to provide a complete and secure platform able to perform breast cancer lesions detection in mammograms. The protocol conducted in the experiment is now detailed:

1) The user authenticates to the front-end server. The authentication system is the one used in the Grid5000 project (see II-A). Communications between the user machine and the front-end server are encrypted using SSL to ensure privacy of the request.
2) The user submits a new mammogram $\mathcal{I}$ to analyze.
3) The controller submits to the storage grid the meta-data of the image $\mathcal{I}$ to select a set of indexes on n images $\{\mathcal{I}_i\}_{0\leq i<n}$ that match the meta-data of $\mathcal{I}$.
4) The images of the set $\left\{\mathcal{I} \cup \{\mathcal{I}_i\}_{0\leq i<n}\right\}$ are anonymized (see §II-C). Then, the farm manager submits n comparison jobs, each of them receiving the images $\mathcal{I}$ and $\mathcal{I}_i$ as inputs and computing the similarity score r_i (see §II-F).
5) A certification process is launched to validate the similarity computations (see §II-B).
6) The farm manager submits sorting jobs to execute a fault-tolerant extension of the bitonic algorithm (§II-G).
7) The sorting process is certified using the result-checking algorithms developed in §II-B. This produces a table T containing the sorted scores together with indexes on the corresponding images $\mathcal{I}_i$.
8) Only the first results are likely to interest the user. Consequently, only the 10% first entries of T are returned.

This complete architecture has been successfully deployed on Grid5000 where unsafe resources have been simulated to validate the approach. For this experiment, we only had a small database of mammograms (for legal reasons, it appears difficult to access medical images). Yet we hope that the encouraging results presented in this paper will permit an access to a wider set of mammograms: As mentioned in the introduction, we negotiate access to a distributed database on EGEE.

IV. Conclusion & Future Works

This paper presents an illustration of a federated research within the RAGTIME project. The specialities of the respective authors have been combined to provide a robust and secure architecture, able to process medical images for assisted diagnosis. The infrastructure is reachable from any machine connected to the Internet, therefore improving the mobility of care practitioner susceptible of using it. He gains a quick and easy access to results, even from its desk. In the context of this article, we considered an application of breast cancer lesions detection in mammograms (even if this infrastructure can be extended to any kind of medical image processing). The complete architecture has been successfully deployed and validated on the Grid5000 large scale infrastructure even if we only have a small database of images. Having access to a bigger database will make it possible to provide significant experimental results. A current work in progress consists in designing a graphical client to illustrate each step of the application described in §III. Future works include the integration of the access to an EGEE database of medical images and the use of smartcards for authentication in step (1) of figure 4.

References

[1] A. Krings, J.-L. Roch, S. Jafar, and S. Varrette, "A Probabilistic Approach for Task and Result Certification of Large-scale Distributed Applications in Hostile Environments," in *Proceedings of the European Grid Conference (EGC2005)*, ser. LNCS 3470, S. Verlag, Ed., LNCS. Amsterdam, Netherlands: Springer Verlag, February 14–16 2005.

[2] S. Jafar, T. Gautier, A. W. Krings, and J.-L. Roch, "A checkpoint/recovery model for heterogeneous dataflow computations using work-stealing." in *Euro-Par*, 2005, pp. 675–684.

[3] S. Varrette, S. Georget, J. Montagnat, J.-L. Roch, and F. Leprevost, "Distributed Authentication in GRID5000," in *LNCS OnTheMove Federated Conferences - Workshop "Grid Computing and its Application to Data Analysis (GADA'05)"*, ser. LNCS 3762, R. Meersman and al., Eds. Agia Napa, Cyprus: Springer Verlag, November 1 2005, pp. 314–326.

[4] L. F. G. Sarmenta, "Sabotage-Tolerance Mechanisms for Volunteer Computing Systems," in *ACM/IEEE International Symposium on Cluster Computing and the Grid (CCGrid'01)*, Brisbane, Australia, Mai 2001.

[5] N. Dagorn, N. Bernard, and S. Varrette, "Practical Authentication in Distributed Environments," in *IEEE International Computer Systems and Information Technology Conference (ICSIT'05)*, IEEE, Ed., Sheraton Hotel, Alger, July 19–21 2005.

[6] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *International J. of Supercomputer Applications and High Performance Computing*, vol. 11, no. 2, pp. 115–128, Summer 1997.

[7] S. Jafar, S. Varrette, and J.-L. Roch, "Using Data-Flow Analysis for Resilence and Result Checking in Peer to Peer Computations," in *IEEE DEXA'2004 - Workshop GLOBE'04: Grid and Peer-to-Peer Computing Impacts on Large Scale Heterogeneous Distributed Database Systems*, IEEE, Ed., Zaragoza, Spain, September 2004, pp. 512–516.

[8] A. W. Krings, J.-L. Roch, and S. Jafar, "Certification of large distributed computations with task dependencies in hostile environments," in *IEEE Electro/Information Technology Conference , (EIT 2005)*, IEEE, Ed., Lincoln, Nebraska, May 2005.

[9] L. Seitz, J. Montagnat, J. M. Pierson, D. Oriol, and D. Lingrand, "Authentication and Authorization Prototype on the μgrid for Medical Data Management," in *From Grid to Healthgrid, Proceedings of Healthgrid 2005*. Oxford, UK: IOS Press, April 2005, pp. 222–233.

[10] A. Shamir, "How to Share a Secret," in *Communications of the ACM*, vol. 22, 1979, pp. 612–613.

[11] L. Seitz, J. M. Pierson, and L. Brunie, "Encrypted Storage of Medical Data on a Grid," *Methods of Information in Medicine*, vol. 44, no. 2, pp. 198–201, February 2005.

[12] L. Seitz, J. Montagnat, J.-M. Pierson, D. Oriol, and D. Lingrand, "Authentication and autorisation prototype on the microgrid for medical data management," in *HealthGrid05*, Oxford, United Kingdom, Apr. 2005.

[13] J. Montagnat, V. Breton, and I. Magnin, "Partitionning medical image databases for content-based queries on a grid," *Methods of Information in Medicine*, vol. 44, no. 2, pp. 154–160, 2005.

[14] J. Montagnat, F. Bellet, H. Benoit-Cattin, V. Breton, L. Brunie, H. Duque, Y. Legré, I. Magnin, L. Maigne, S. Miguet, J.-M. Pierson, L. Seitz, and T. Tweed, "Medical images simulation, storage, and processing on the european datagrid testbed," *Journal of Grid Computing*, vol. 2, no. 4, pp. 387–400, Dec. 2004.

[15] T. Glatard, J. Montagnat, and I. Magnin, "Texture based medical image indexing and retrieval: application to cardiac imaging," in *Proceedings of ACM Multimedia 2004, workshop on Multimedia Information Retrieval (MIR'04)*, New York, USA, Oct. 2004.

[16] T. Tweed and S. Miguet, "Automatic detection of regions in interest in mammographies based on a combined analysis of texture and histogram," in *International Conference on Pattern Recognition*, Quebec City, Canada, Aug. 2002.

[17] K. E. Batcher, "Sorting Networks and their Applications," in *Proc. AFIPS Spring Joint Computer Conference*, vol. 32, 1968, pp. 307–314, http://www.cs.kent.edu/~batcher/sort.ps.

[18] M. Ajtai, J. Komlos, and E. Szemeredi, "An $O(n \log n)$ sorting network," in *Proc. of the 15th Annual ACM Symposium on Theory of Computing*, Boston, April 1983, pp. 1–9.

[19] S. Assaf and E. Upfal, "Fault Tolerant Sorting Networks." *SIAM J. Discrete Math.*, vol. 4, no. 4, pp. 472–480, 1991.

[20] T. Leighton, Y. Ma, and C. G. Plaxton, "Breaking the $\mathcal{O}(n \log^2 n)$ Barrier for Sorting with Faults," *Journal of Computer and System Sciences*, vol. 54, no. 2, pp. 265–304, 1997.