ARAMIS: A Remote Access Medical Imaging System

David Sarrut and Serge Miguet
{dsarrut,miguet}@univ-lyon2.fr

Laboratoire ERIC - Université Lumière Lyon 2
Bat. L - 5 av Pierre Mendès-France
69676 Bron Cedex - France

Abstract

In Hospital services, practitioners need to access and study large data volume (3D images) with help of specific, parallel, high performance processing tools. This work describes the ARAMIS platform (A Remote Access Medical Imaging System), which allows transparent remote accesses to parallel image processing libraries. Such system is based on a communication protocol which takes as input parallel libraries (written in C) and leads to Java objects, which can be combined easily. The end-user application is thus a Java applet, allowing any common workstation to activate, in a convivial way, time-consuming parallel processing.

1 Introduction

As a part of a project called "Health and HPC" whose goal is to bring High Performance (HP) Computing resources in Hospital services, our team focuses on HP image processing tools acting on large data volumes (such as 3D images).

In Hospital services, 3D images databases are distributed over several linked services (cardiology, radiology), potentially accessible from anywhere through a local network. Such images represent a considerable volume of distributed data which must be acceded and visualized by the practitioners with specific medical image processing tools.

For some years, the research community on parallel algorithm for image processing has developed a large number of powerful algorithms and methods, dedicated to MIMD architecture (from fine-grained to mid-grained parallelism, including clusters of workstations). However, these tools are often very optimized and fully efficient at the expense of a limited accessibility, either for end-user practitioners or programmers who want to combine several tools and build large and accessible applications. Moreover, powerful parallel machines are costly and few of them could be present in a same Hospital.

Metacomputing techniques, which allow a same application to accede and use various remote resources, appear to be especially well suited for such a situation [FK98]. In our case, the resources are: large amount of data, powerful image processing libraries, and parallel machines. In order to bring convivial access to such resources, we must take into account the following constraints: the amount of data circulating on the network must be low, the machines are heterogeneous (both hardware architecture and system) , the libraries are written with C language (not object-oriented) and security access to the data must be planned.

Hence, we propose a system which allows to easily build (in Java) end-user applications with friendly GUI and with transparent access to remote resources. This system is called ARAMIS (A Remote Access Medical Imaging System). Many modifications has occurred from the beginning of the project [Sar98]. Especially, the whole communication protocol has been fully rewritten in an object-oriented way (section 4). Moreover, our tool allows to integrate the parallel libraries in a much simpler way and the system can now manage dynamically several remote servers. The system can be summarized with three main points:

- in order to avoid overflowing the network, the large data (3D images) stay either on data storages or on the machines dedicated to the HP medical image processing. The client only receives 2D images, displayable on any common workstation (with low memory and weak computing capacities).
- the top-level application is implemented in Java, because of the applet mechanism (Web access), and of the object-oriented core which allows transparent use of remote objects.
- a Meta Object Protocol gives programmers an easy control on remote data, and a whole access to the HP libraries acting on these data.

This paper is organized as follows. The section 2 refers to some relationship between related works and our system. In the next section, we present an overview of ARAMIS, and more details on the architecture in section 4. We conclude in section 5 by future works and snapshots of the current prototype.

2 Related Works

Taken as a whole, our approach follows the classical three layers architecture: a transport protocol, an Meta Object Protocol (MOP) which supports the distributed object mechanism and a final API (Application Programming Interface), which is the high-level series of functions that programmers can use to build end-user application. In order to refer our project through the Metacomputing community, we briefly present similarities and differences with projects or general purpose communication schemes which have been already developed.

The NetSolve project [CD97] aims at bringing remote access to scientific computing libraries. Even if our goal is also to offer access to remote libraries, the nature of the data (images) and the processing tools are very different. Our approach is specifically oriented to medical image processing and is not suited for other general data types. Moreover, more general projects such as NetSolve or Globus [CFKK98] concern Internet-wide computing, whereas in our approach, the set of remote resources is acceded through a local network in order to keep efficiency.

[vLSI+99] described a software architecture, built over the Globus technology [CFKK98], which (among other features) enables from generation to visualization of Computed Microtomography images. The system deals with Giga Bytes data volumes and parallel algorithms (reconstruction of microtomographic datasets from numerous slices). However, the visualization part of the project is built over hardware-optimized libraries. As our requirements are to provide accessibility to HP visualization tools from *any* common workstation (via Web browser), this approach does not seems to be suitable in our case (even if such hardware acceleration could also be used in the libraries standing on the servers, see section 3.1).

There are two kinds of communication modules in our system. The first is used in most of the parallel tools and is related to the classical PVM or MPI message passing paradigm (section 3.2). The second concerns client-server communication and deals with high-level remote object management. A Java module on server's side has first being envisaged. The RMI (Remote Method Invocation) technology provided with the JDK was used in junction with the JNI (Java Native Interface) library, but it leads to decrease performance and to add a supplementary amount of code in server side. Because of the heterogeneity of both used languages and involved machines, we think to investigate the use of the CORBA technology in future work. However, such a choice would imply several issues (most of our libraries are not oriented-object, does all the browsers include ORB ?) and a large amount of code to be added to the server. Other general framework such as the IceT project [GS97] or Nexus [FTT97] (the communication module of the Globus project previously cited), are presently under investigation in order to replace the non object-oriented part of the server (section 4).

3 ARAMIS overview

In this section, we present the overall aspects of the framework, more details are given in the next section.

3.1 Hardware architecture

In Hospital services, we consider several databases storages (corresponding to different services or images acquisition devices), and a few number of powerful machines (which could be real multi-processor machines, or NOWs: Networks Of Workstations). The parallel machines will act as powerful graphics computing and rendering servers. A server stands also for databases management, and makes use of the parallel server through several high-performance mechanisms [BM98].

The databases sites and the parallel machines are linked by a high-speed network, independent of the network which links the end-user machines with the computing resources. Hence, this scheme defines two levels of network (see figure 3.1): the first one (between databases and servers) supports transport of large volume of data, and the second one could be classical low bandwidth network, because it only sends 2D images from parallel machines to practitioner's workstation.

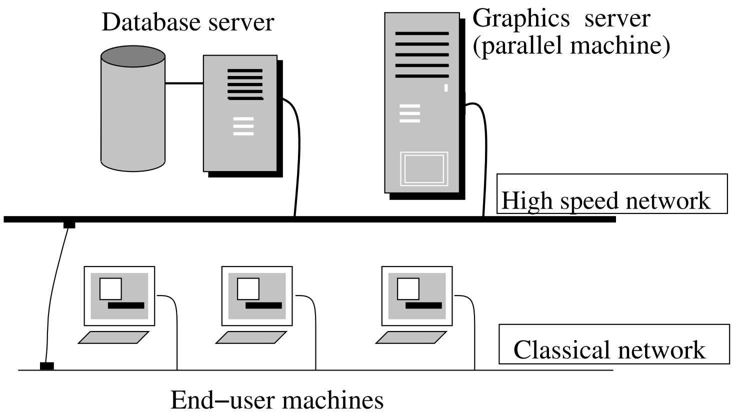

Fig. 1. Two levels of network

Special attention have been paid to make use of existing materials, and to propose cheap hardware configuration.

3.2 Java access to remote HP image processing libraries

We manage a set of libraries, each of them consisting in several image processing algorithms. These libraries has been developed for some years (in C language) and with the parallel communication library called PPCM [CBF$^+$92], which allows to compile the code for any defined target platforms (PVM, MPI, or parallel architecture such as Cray T3E or Intel Paragon). The range of image processing covered by these tools consists in time-consuming tasks requiring high computing capabilites (both memory and CPU usage). For instance: parallel Volume Rendering [LM92], parallel (real-time) surface extraction with the Marching-Cubes algorithm [MN95], parallel Z-Buffer [CLM95], (sequential) optimized 3D volumes registration [SM99] and so on.

According to a set of mechanisms embedding the low-level communication details (described in section 4), we can access to these libraries with a Java applet. The application is thus encompass in a Web browser or can also be used as a standalone application. It should be notice that, by this way, the practitioner do not have to change his usual work environment nor his workstation.

3.3 Data flow

The large 3D volumes of data are *never* transfered to the user's workstation. The applet only receives 2D images, resulting from a remote processing. This is done in a transparent way for the practitioner which seems to have a powerful parallel machine in front of him. For example, for a simple visualization purpose, only 2D (compressed) slices of volumes are sending on request (see figure 3), when acting on a slider. Such a process is fast enough to provide interactive displaying on an Ethernet local network.

Moreover, we consider three different kinds of tasks:

– the low resource-consuming tasks are done on user's machine (colors enhancement for instance, see Colormap Editor figure 3).
– the high resource-consuming tasks are done remotely and using remote data. Typically, it concerns processes which would normally take several minutes to complete on a monoprocessor workstation, and which are accomplished in few seconds with this system. For example, figure 5 shows the result of a Volume Rendering (with transparency effects) algorithm which is displayed on client machine within few seconds.
– however, some *interactive* tasks (real-time 3D visualization of millions of polygons for instance) would only be displayed at a low frame rate due to the network overhead. Thus, instead of overflowing the network with such particular tasks and in order to keep usability for very weak end-user workstation, we advocate the use of a two-steps method. The choice of the spatial position of the 3D object is done locally through a simplified interface, and the full-resolution and true-color rendering is done remotely on user's request (or when mouse releases the virtual track-ball).

This approach allows to spread tasks on remote server or local machine according to its nature. Hence, the server is not bother by a considerable number of different tasks which can be done locally and can thus quickly serve the clients. The next section presents the underlying communication protocol.

4 The Meta Object Protocol

This section deals with the set of mechanisms involved in the communication management between the end-user application (Java), and the remote parallel libraries (C language). The following protocol aims at providing, not a totally transparent management to remote resource, but a control as simple as possible in order to keep the full efficiency of the HP tools. We first describe the steps of connection establishment between the applet and the remote parallel server, and then present the MOP used for creating Java objects which can then be used by any application.

4.1 Client/Server relation

Because of the particular nature of our environment, we are concern with a non-symmetric client/server relation because the servers do not have to act on the client part, except for very limited tasks (sending 2D images, or 3D images characteristics). On the other hand, client applications can only activate two kinds of tasks: loading data (remotely) and activate processing on such data. Hence, we do not provide mechanisms which allow to create remote object symmetrically from server to client and from client to server (such as the IceT environment[GS97] or the Do! project [LP98]).

When the applet is started on the client machine, a connection to a process running on the server (called `Aramis_Daemon`) is established. This connection allows to ask for the different servers available, each of them providing a different set of processing. We distinct presently three servers: a parallel one (dedicated to time-consuming image processing), a sequential one (for processing such as slices delivering or optimized volume registration [SM99] which is not yet parallelized), and a database server which will allow to select and transport the images between servers.

Each connection to a server provide a set of objects embedding the communication details and allowing remote access and control.

4.2 Providing simple objects to control remote resources

The proposed protocol is build with classical object-oriented methods, but is adapted for our purpose and particularities with the goal of reducing as most as possible the network overhead. Our system stands between two kind of developers: those who provide HP parallel tools, and those who build end-user applications dedicated to the specific needs of practitioners. The former should depose their libraries on a repository and provide as few as possible supplementary code to integrate their tools into the system. The latter should only manipulate some high-level objects which interact transparently with the remote processes.

At the lowest-level of the protocol (Transport layer), we use TCP/IP as data transport with the classical socket interface, because of the integration with the C language and the Unix architecture used on most of the parallel machines. At a higher level (Remote Reference layer), we choose to developed a specific purpose *stub-skeleton* scheme (figure 4.2).

On server's side, several entry points are extracted from each libraries and are inserted into the system by adding simple functions. This step is presently done manually by the library's developer but require very few lines. Because of the very limited set of data types (3D matrix of voxels [Mig92], or lists of polygons), a true IDL (Interface Definition Language) is not yet provided.

Moreover, parallel image processing techniques often need to spread data over the different processors for efficient load-balancing. Thus, between two executions of different treatments, data must be redistributed. This is expected to be done automatically with help of the *ParList* data redistribution algorithm [FMP98], which allows well balanced workload while keeping the overcost of the redistribution at a small value. Hence, before each parallel algorithm execution, a simple and automated call to this library will allows to distribute the data in an efficient way for the process. Such a strategy is currently not yet fully integrated into ARAMIS, but we actively pursuing this goal.

On client's side, simple object (which act as *stub*) are then provided. They embed the communication details and allow to activate and wait (asynchronously

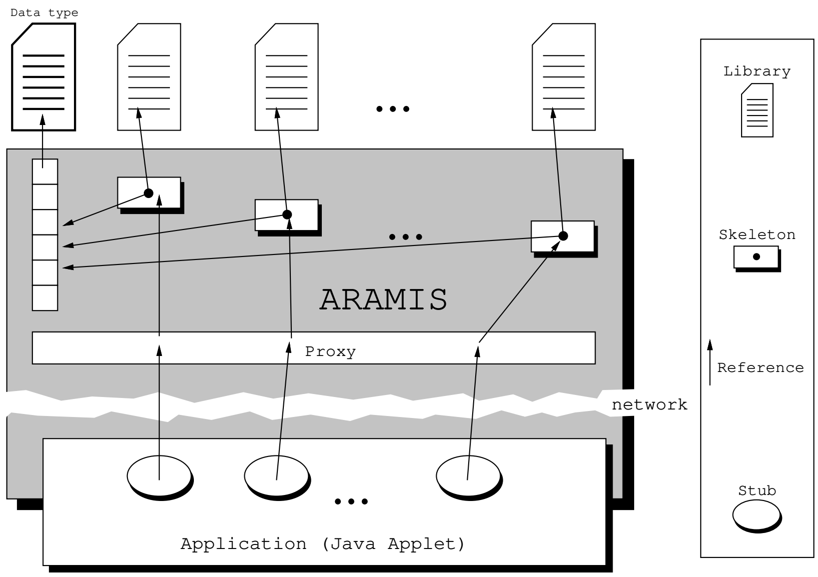

Fig. 2. Overview of ARAMIS. The grey part depicts the hidden protocol which allows to connect Java objects to remote libraries

with use of Java thread) for remote processing. The data remote references on client's side are also lightweight objects.

Such communication scheme is thus a specific purpose implementation of classical mechanisms. The next step will be to compare it with available ORB included in some browsers.

5 Conclusion

We have proposed in this work a system called ARAMIS (A Remote Access Medical Imaging System) which allows practitioners to activate HP medical images processing with a friendly GUI in their Web browser. The system is built with a communication protocol taking as input parallel libraries (written in C) and leading to Java objects, which can be combined in order to build end-user applications.

A previous prototype of ARAMIS has already being presented in [Sar98], but the core of the system has being fully rewritten in order to integrate classical object-oriented metacomputing techniques (*stub-skeleton*, Remote Reference). This project is still in development and integration of all the parallel libraries are in progress. Nevertheless, in order to show the feasibility of this approach, a prototype is currently functional and is daily used in our laboratory. In order to demonstrate the suitability of the system, a limited applet is accessible through our Web page[1]. It allows to activate remote loading of 3D volumes, and displaying 2D slices on request.

Several issues are still under investigation. For instance, we plan to study the system described in [GGMS98] which aims at providing an interface from Java to C libraries. Such an approach could be used in order to produce automatically the *skeleton* part of our design. However, data types conversion (a critical issue in image-purpose applications) seems to be a difficult task if we want to keep full efficiency in the system. Moreover, the remote I/O used in our approach are rather basic and we think other paradigms for access to distributed data should be investigated [FKKM97].

As futur improvements, we investigate the use of a fully Java communication protocol (both client and server's side), with help of general framework such as [LP98]. Moreover, reflective protocol such as Reflective RMI [TTK98] seems to be a interesting way which should allows us to dynamically build remote reference object. In the short term, we will start a collaboration with researchers on parallel database systems [BM98] in order to make HP algorithms for medical image mining (parallel request optimizer, distributed execution support and so on) accessible with ARAMIS.

Acknowledgment

This work is supported by the Région Rhône-Alpes under the grant "*Santé et Calcul Haute-Performance*" (Health and High Performance Computing).

Snapshots

(see next pages)

References

[BM98] L. Brunie and E. Mayer. Distributed Systems and Databases. In *4th International Euro-Par Conference, Southampton*, volume 1470. LNCS, September 1998.

[CBF+92] H.P. Charles, O. Baby, A. Fouilloux, S. Miguet, L. Perroton, Y. Robert, and S. Ubéda. PPCM: A Portable Parallel Communication Module. Technical Report 92-04, LIP-IMAG, ENS-Lyon, 46 allée d'Italie, 69364 Lyon CEDEX 07, 1992.

[CD97] H. Casanova and J. Dongarra. NetSolve: A Network-Enabled Server for Solving Computational Science Problems. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(3):212–223, 1997.

[CFKK98] K. Czajkowski, I. Foster, N. Karonis, and C. Kesselman. A Resource Management Architecture for Metacomputing Systems. *Lecture Notes in Computer Science*, 1459:62–81, 1998.

[1] http://eric.univ-lyon2.fr/~dsarrut/aramis

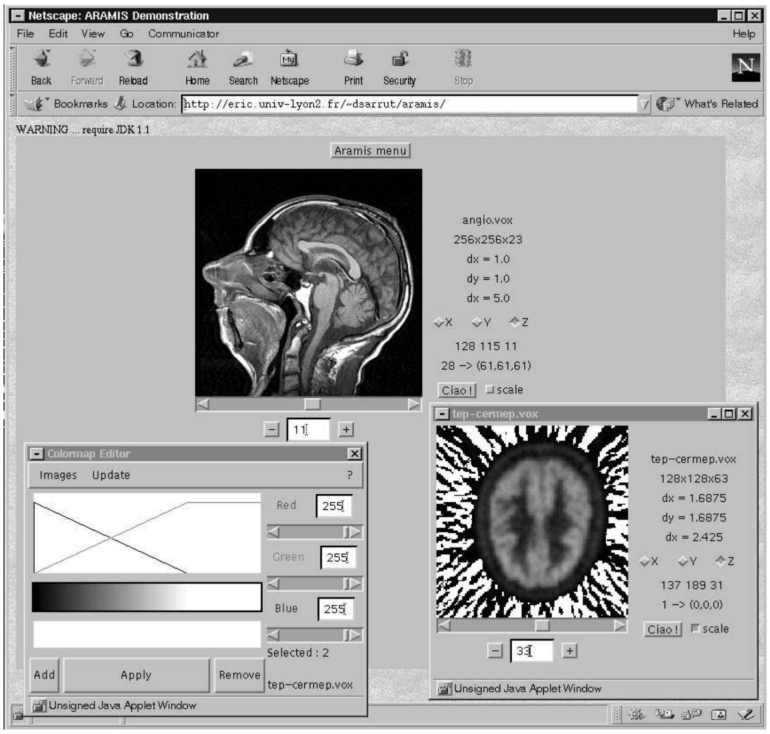

Fig. 3. The Java applet embedding a Slice deliver: the 3D volumes remain on server, the 2D slices are sent to the client on request. The two floating windows are : a Colormap Editor and an other Slice deliver.

[CLM95] H.P. Charles, L. Lefevre, and S. Miguet. An Optimized and Load-Balanced Portable Parallel ZBuffer. In *SPIE Symposium on Electronic Imaging: Science and Technology*, 1995.

[FK98] I. Foster and C. Kesselman. The globus project: A status report. In *IPPS/SPDP '98 Heterogeneous Computing Workshop*, pages 4–18, 1998.

[FKKM97] I. Foster, D. Kohr, R. Krishnaiyer, and J. Mogill. Remote I/O: Fast Access to Distant Storage. In *Workshop on I/O in Parallel and Distributed Systems (IOPADS)*, pages 14–25, 1997.

[FMP98] F. Feschet, S. Miguet, and L. Perroton. ParList: a Parallel Data Structure for Dynamic Load Balancing. *Journal of Parallel and Distributed Computing*, 51:114–135, 1998.

[FTT97] I. Foster, G. K. Thiruvathukal, and S. Tuecke. Technologies for Ubiquitous Supercomputing: a Java interface to the Nexus communication system. In G. C. Fox, editor, *Java for Computational Science and Engineering — Simulation and Modeling: Workshop — Syracuse, KS*, volume 9:6, pages 465–476. John Wiley and Sons, December 1997.

[GGMS98] V. Getov, P. Gray, S. Mintchev, and V. Sunderam. Multi-Language Programming Environments for High Performance Java Computing . In *Java for High Performance Network Computing.* EuroPar'98, September 1998.

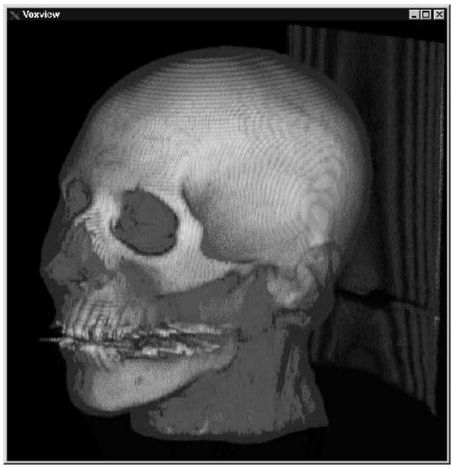

Fig. 4. Result of a remote Volume Rendering

[GS97] P.A. Gray and V.S. Sunderam. IceT: Distributed Computing and Java. In *ACM Workshop on Java for Science and Engineering Computation*, volume 9:11, November 1997.

[LM92] J.J. Li and S. Miguet. Parallel Volume Rendering of Medical Images. In Q. Stout, editor, *EWPC'92: From Theory to sound Practice*, pages 332–343, Barcelone, 1992.

[LP98] P. Launay and J.L. Pazat. Generation of Distributed Parallel Java Programs. Technical Report RR-3358, INRIA, February 1998.

[Mig92] S. Miguet. Voxcube: a 3D imaging package. Technical Report 92-05, Ecole Normale Supérieure de Lyon, 1992.

[MN95] S. Miguet and J.M. Nicod. An Optimal Parallel Iso-Surface Extraction Algorithm. In *Fourth International Workshop on Parallel Image Analysis (IWPIA'95)*, pages 65–78, December 1995.

[Sar98] D. Sarrut. ARAMIS: an "on line" Parallel Platform for Medical Imaging. In H. R. Arabnia, editor, *International Conference on Parallel and Distributed Processing Technique and Applications*, pages 509–516. CSREA Press, July 1998.

[SM99] D. Sarrut and S. Miguet. Fast 3D Images Transformations for Registration Procedures. In *10^{th} International Conference on Image Analysis and Processing*. IEEE Comp. Society Press, September 1999. To appear.

[TTK98] G. K. Thiruvathukal, L. S. Thomas, and A. T. Korczynski. Reflective Remote Method Invocation. In *ACM 1998 Workshop on Java for High-Performance Network Computing*, February 1998.

[vLSI+99] G. von Laszewski, M.H. Su, J. A. Insley, I. Foster, J. Bresnahan, C. Kesselman, M. Thiebaux, M. L. Rivers, S. Wang, B. Tieman, and I. McNulty. Real-Time Analysis, Visualization, and Steering of Microtomography Experiments at Photon Sources. In *Ninth SIAM Conference on Parallel Processing for Scientific Computing*, April 1999.