# Distributed Computing in Algebraic Topology: first trials and errors, first programs. *

Andrés, M. Pascual, V. Romero, A.† Rubio, J.

Departamento de Matemáticas y Computación.
Universidad de La Rioja.
(email:{miriam.andres, mvico, ana.romero, julio.rubio}@dmc.unirioja.es)

Nowadays, Internet appears as a suitable tool for performing scientific computations in a distributed collaborative way. One of the fields where this idea can be applied is that of Symbolic Computation. Computer Algebra packages are being extended to interconnect them. In fact, some Computer Algebra systems are already capable of performing distributed computing. An example is Distributed Maple [7], which is an environment for executing parallel computer algebra programs on multiprocessors and heterogeneous clusters.

With this example in mind, it is clear that this trend could be explored in any other Symbolic Computation system, in particular in systems devoted to algorithmic Algebraic Topology. The leader systems in this field are EAT [3] and Kenzo [4]. These systems, created by Sergeraert, can be used to compute homology groups of infinite topological spaces, namely loop spaces. Both systems are written in the Common Lisp programming language and have obtained some results (specifically, homology groups) which had never been determined before using neither theoretical nor computational methods. Since the programs are very time (and space) consuming, it is obvious that they are specially well placed to try making them distributed.

EAT and Kenzo are written in the Common Lisp programming language, but Java is the core language in the net-centric mainstream technology. That is the reason why we first thought of reconstructing the systems in the Java language. As an intermediate step, we rebuilt (some fragments of) EAT in the statically typed functional programming language ML [6]. Thus, we have implemented two prototypes to perform computations in Algebraic Topology in ML and Java respectively.

The following step we undertook was to interoperate between these systems through the net. Our first attempt was to use MathML (Mathematical Markup Language) [5] (an XML dialect which was the first standard to represent mathematics) as exchange language, but it turned out to be not adequate as the elements of the algebraic structures that EAT

*Partially supported by Ministerio de Ciencia y Tecnología, project TIC2002-01626 and by Comunidad Autónoma de La Rioja, project ACPI-2002/06.

†Supported by a FPU research grant, Ministerio de Educación, Cultura y Deporte.

works with are more complicated than those of MathML. Thus, we were forced to define our own DTD (see [8]), and we defined it as a proper extension of the MathML DTD for further compatibility.

In a previous work, we have implemented a prototype that allows the three systems previously mentioned (the Common Lisp, ML and Java versions) to interchange XML documents (among them and also with themselves), as a first step towards collaborative computing (see [1]). For the moment, the prototype works in a local, non-distributed way.

In another paper, the features of some Symbolic Computation Systems for Algebraic Topology, which are relevant to the objective of rebuilding them as distributed systems, were detected. These characteristics are:

1) They are very time and space consuming.
2) They are written in Common Lisp.
3) They make use of higher order functional programming (to deal with infinite data structures).
4) They are organised in two layers of data structures.
5) They have been capable of computing results up to now unknown (stressing the importance of high reliability and of partial correctness proofs).

Our first trials towards distributed computing have been to use our XML format as exchange language and to try working with CORBA [2], a middleware that allow distributed and client/server programming, especially used with computers connected by a local net. CORBA is based in objects, which have an interface written in IDL (a new interface description language). Our first attempts have confirmed that some naive thoughts on these topics are misleading:

1) XML solves the problem of interoperability. This is obviously an uncomplete picture, because it is necessary to have an infrastructure that allows the interchange between the client and the server. This infrastructure could be, for instance, CORBA.
2) CORBA provides a solution to the problem of distributed interoperability. It is not true, because the IDL interfaces of client and server need to be the same. In this way, although IDL is language independent and can be used to translate to/from a variety of different languages, such as C, C++, Lisp and Java, its use includes the need of an extra design work to unify the client and server interfaces.
3) In an homogeneous environment (i.e., with the same programming language on the client and server sides), CORBA is the adequate solution of the problem. False again, because even if working in a homogeneous environment (with Common Lisp, in our case), the problem of data interchange between client and server reappears.

At this point, the role of XML as interchange language is considered again, providing the necessary "glue" to solve this last problem. More specifically, the interchange can be managed by inserting two string attributes in the object interface which will contain the XML representation of the call to the Kenzo function and the result of the computation (see figure 1). In the poster we will illustrate different programs which have made possible the distributed versions of the local and non-distributed prototypes previously mentioned.

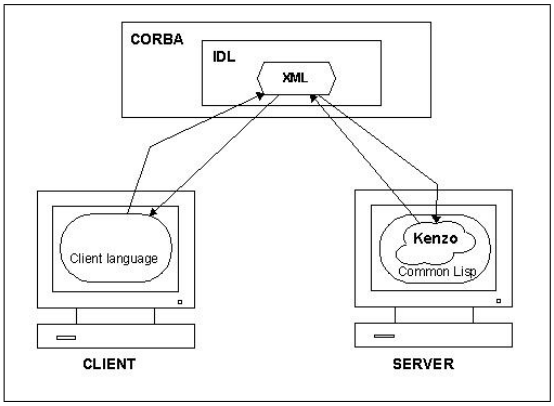


Figure 1: Scheme of the client/server Kenzo application

## References

[1] M. Andrés, F. J. García, V. Pascual, J. Rubio. *XML-Based interoperability among symbolic computation systems*, in Proceedings WWW/Internet 2003, Vol. II, Iadis Press (2003) 925-929.

[2] Object Management Group. *Common Object Request Broker Architecture (CORBA)*. `http://www.omg.org`.

[3] J. Rubio, F. Sergeraert, Y. Siret. *EAT: Symbolic Software for Effective Homology Computation*. Institut Fourier, Grenoble, 1997. `ftp://ftp-fourier.ujf-grenoble.fr/pub/EAT`.

[4] X. Dousson, F. Sergeraert, Y. Siret. *The Kenzo program*, Institut Fourier, Grenoble, 1999. `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/`.

[5] Ausbrooks, R. et al. *Mathematical Markup Language (MathML)* Version 2.0, 2003. `http://www.w3.org/TR/2001/REC-MathML2-20010221/`.

[6] L. C. Paulson. *ML for the working programmer*, $2^{nd}$ edition. Cambridge University Press, 2000.

[7] Schreiner, W. et al. *Distributed Maple: Parallel Computer Algebra in Networked Environments*, in Journal of Symbolic Computation, Vol. 35, n 3 (2003) 205-347.

[8] T. Bray et al (eds). Extensible Markup Language (XML) 1.0 (Second edition), 2003. `http://www.w3.org/TR/REC-xml/`.