A Web-based Metacomputing Problem-Solving Environment for Complex Applications

Ranieri Baraglia[1], Domenico Laforenza[1], and Antonio Laganà[2]

[1] CNUCE-Institute of the Italian National Research Council
CNR Research Campus, Via V. Alfieri 1, 56010 Ghezzano, Pisa, Italy
e-mail:(Ranieri.Baraglia,Domenico.Laforenza)@cnuce.cnr.it
[2] Dipartimento di Chimica, Università di Perugia
Via Elce di Sotto, 8 - I06123 Perugia (Italy)
e-mail:lag@unipg.it

Abstract In this paper a kernel of Problem Solving Environment aimed at managing complex chemical meta-applications based upon an *a priori* simulation of molecular structure and dynamics has been presented. By considering as a case study the simulation of a molecular beam experiment (SIMBEX), a metacomputing environment able to facilitate the SIMBEX execution through the Web has been designed. This choice is due to the rapid and impressive growth of Internet, Java and, Web technologies. The current work focus on the architectural aspects of the implemented environment.

1 Introduction

Modern Computational Sciences increasingly stimulate the development of advanced computing tools because of their need for realistic simulations of complex systems relevant to the modeling of several modern technologies and environmental phenomena. This type of simulations usually needs to include, though not necessarily in a completely rigorous manner, a detailed description of relevant molecular structures and processes. As a result, related computational procedures not only need to be run by coordinating several complementary expertises but also by integrating several extremely powerful computing platforms in a metacomputer system. From this derives the need to build smart and user-friendly Problem Solving Environments (PSE) enabling computational scientists to carry out their investigations without caring about the complexity of the computing platform being used. As defined in literature, a PSE is a computer system that provides all the computational facilities needed to solve a target class of problems. These features include advanced solution methods, automatic and semiautomatic selection of solving procedures, easy incorporation tools of novel approaches [1].

A European COST Initiative has been recently proposed [2] to promote the gathering of research laboratories having complementary expertises in clusters grafted on metacomputer systems (Metalaboratories). This action has been recently approved (D23) and a call for cooperative projects is being issued. These projects should tackle complex modeling problems without conveying in a single location all the required laboratories, programs and pieces of hardware. Our proposal focuses on building a Metalaboratory devoted to the *a priori* simulation of molecular processes, and in particular of crossed molecular beam experiments.

The metacomputing [3,4] approach harnesses different computational resources and uses their aggregate power as if it was contained in a single machine.

From a technological point of view, the rapid and impressive growth of the Internet has generated a rising interest in Web-based parallel computing. In fact, many worldwide projects are focused on the exploitation of the Web as an infrastructure for running coarse-grained distributed parallel applications. In this context, the Web has the capability to become a suitable and potentially infinite scalable metacomputer for parallel and collaborative work as well as a technological key to create a pervasive and ubiquitous grid infrastructure [5,7].

As a case study, we should consider here the simulation of crossed molecular beam experiments whose (on a small scale) prototype numerical procedure (SIMBEX) has already been discussed in literature [8].

Aim of the present paper is to briefly describe the main features of the simulation and the characteristics of the software tools developed to facilitate the SIMBEX execution on a metacomputer through the Web. These tools are designed to supply a completely transparent support to the user who does not have to care about the localization and the allocation of computing resources. All the needed functionalities were implemented on a properly extended Web server using, whenever possible, standard tools. In particular, use of the Java Servlet [9] and Directory Service facilities of LDAP [10,11] have been made. Moreover, a modular design has been adopted to guarantee an easy maintenance and extendibility of the product.

The paper is articulated as follows. In Section 2 is given a short description of SIMBEX. Section 3 focuses on the architectural aspects of the metacomputing environment. Related work on Web-based metacomputing environments is presented in Section 4. Finally, we summarize our work in Section 5.

2 A Short Description of SIMBEX

SIMBEX is a computational procedure based on *a priori* calculations of structures and processes of molecular systems. The procedure is articulated into several modules derived from the theoretical approach to the problem (see Figure 1).

Each module consists of alternative or coordinated computer codes which accomplish particular tasks. In particular, in module I the construction of the potential energy surface is performed (see Figure 2). This procedure may be bypassed when the potential energy surface is already available or used "on the fly" during dynamical calculations when a direct approach is chosen. If this step is not bypassed then the level of accuracy of *ab initio* calculations, the number and location of points to be considered, the fitting of calculated *ab initio* values to a given functional form have to be performed.

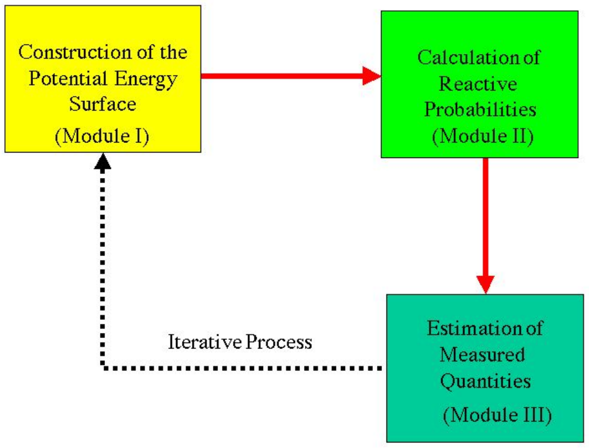
Construction of the Potential Energy Surface (Module I)
Calculation of Reactive Probabilities (Module II)
Estimation of Measured Quantities (Module III)
Iterative Process

Figure1. SIMBEX: Computational Control Flow

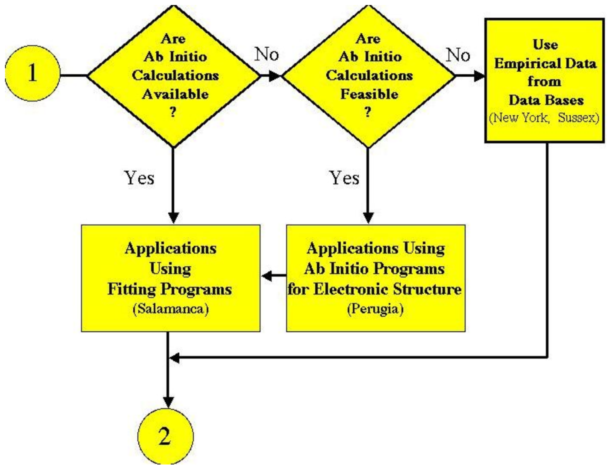
1
Are Ab Initio Calculations Available ?
No
Are Ab Initio Calculations Feasible ?
No
Use Empirical Data from Data Bases (New York, Sussex)
Yes
Yes
Applications Using Fitting Programs (Salamanca)
Applications Using Ab Initio Programs for Electronic Structure (Perugia)
2

Figure2. SIMBEX: Module I Control Flow

In module II dynamics calculations are carried out (see Figure 3). These calculations too can be performed at different levels of accuracy. For small molecules it is possible to perform exact quantum dynamical calculations that can be either of the time dependent or of the time independent type. When considering larger molecules, approximations need to be introduced. This may consist of dynamics constraints leading to a dimensionality reduction in quantum calculations, or of a mixing of quantum and classical techniques, or of a use of pure classical methods.

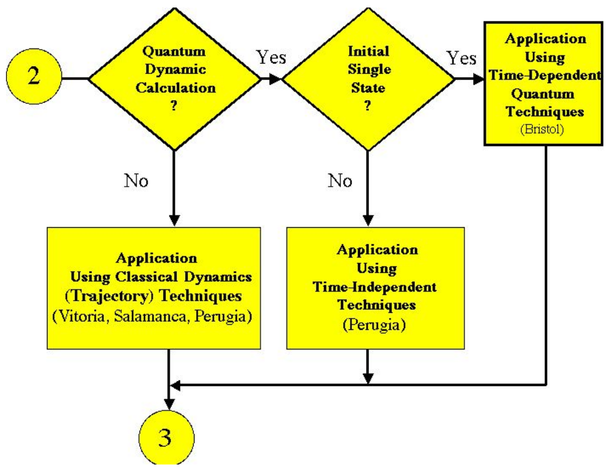
2
Quantum Dynamic Calculation ?
Yes
Initial Single State ?
Yes
Application Using Time-Dependent Quantum Techniques (Bristol)
No
No
Application Using Classical Dynamics (Trajectory) Techniques (Vitoria, Salamanca, Perugia)
Application Using Time-Independent Techniques (Perugia)
3

Figure3. SIMBEX: Module II Control Flow

Finally, in module III (see Figure 4), when scattering matrix elements or state to state probabilities have already been calculated, an averaging over unobserved variables needs to be made to reproduce experimental properties and distributions.

More detailed information about SIMBEX can be found in [8,12].

3 The Metacomputing Environment: Architectural Aspects

To implement SIMBEX on a Web-based metacomputer platform we have designed a 3-tier architecture having the following components:

- Client side: a Web browser;
- Middleware: Web servers exploiting Java Servlets and Lightweight Directory Accesss Protocol (LDAP) functionalities;
- Back-end: the ensemble of computing resources.

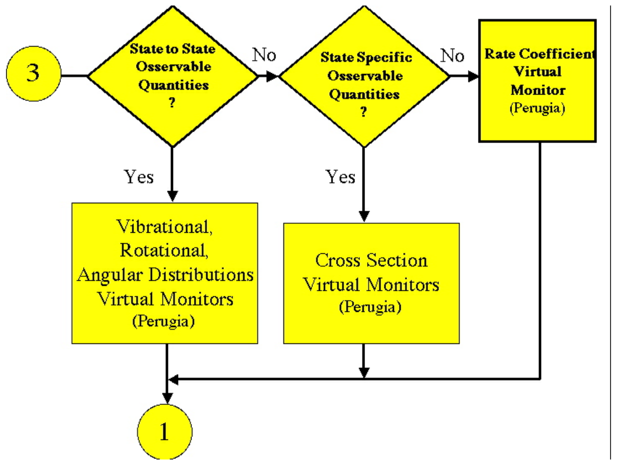
3
State to State Observable Quantities ?
No
State Specific Observable Quantities ?
No
Rate Coefficient Virtual Monitor (Perugia)
Yes
Yes
Vibrational, Rotational, Angular Distributions Virtual Monitors (Perugia)
Cross Section Virtual Monitors (Perugia)
1

Figure4. SIMBEX: Module III Control Flow

Figure 5 shows the architectural scheme singling out the key interactions among the mentioned components. The user, after being authorized when accessing the system, is offered a choice of applications available on the back-end. Next step deals with the handling of input data. In order to satisfy the requests of the users, the server makes use of the LDAP functionalities to localize available computing resources capable to provide requested services. LDAP provides information about the computing resources by accessing to a Directory Information Tree (DIT). The tree is made up of entries which represent the computing resources by a group of attributes. After collecting related information, the server activates a remote execution of the application on the selected machine. When the execution is completed, results are passed to the server that forwards them to the Client.

3.1 The Client Side

The client is made of a Web browser representing the graphical interface driving the user in the selection of the required application, inputting the necessary data and collecting the results. HTML forms ensure the interaction of the user with the Web server: they activate the execution of the Java servlet that corresponds to the requested action (application selection, data input, etc.). Obviously, the address of the Web server providing the service has to be known in advance to the user. The initial page allows the specification of the *userid* and of his *password* which implies also the process of crediting a user (see Figure 6).

This implies the transfer on the network of private information that could be made using HTML forms. In order to check the integrity of the information transmitted we use a Java applet that implements a HMAC [15] mechanism which exploits the iterative cryptographic MD5 hash function.

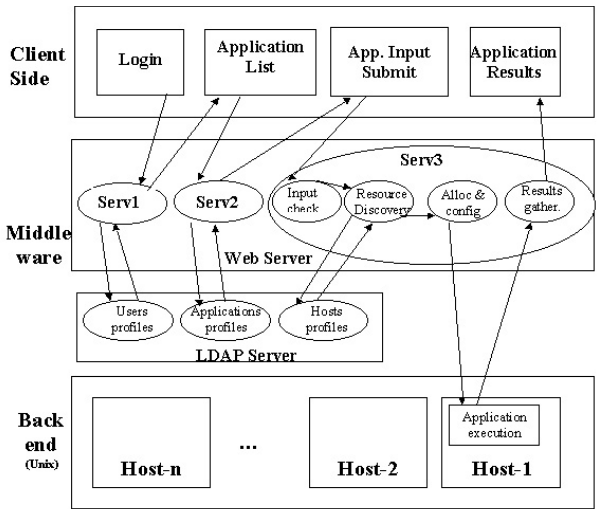

Figure5. The Architectural Scheme Singling out the key Interactions among the System Components.

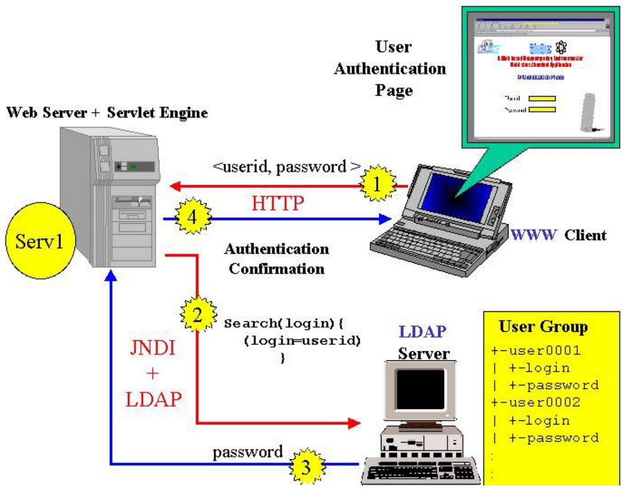

Figure6. User Authentication Process.

In order to control the access to the system's resources, it is possible to define different user profiles according to predefined politics. This has been implemented using LDAP.

After authentication, the user is offered a list of applications that can be run on the machines belonging to the back-end system (see Figure 7). The application selection can be performed by clicking on the hyperlink related to the application.

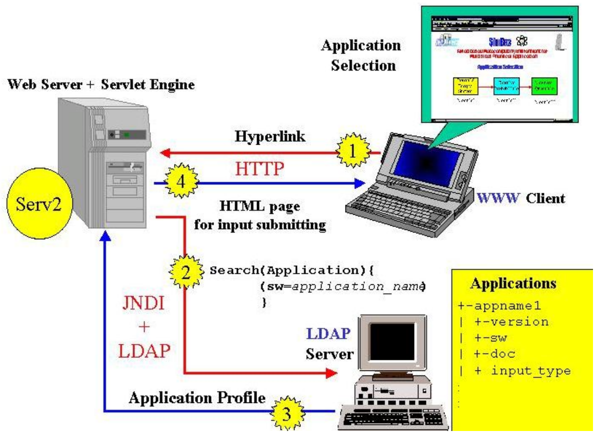

Figure7. Application Selection Process.

Each application has an associated profile describing its computational characteristics (requirements): e.g., name, version, documentation available, type of input required, sources of data, etc. The application profile is stored in the LDAP server that is searched by the Web server in order to drive the input process. According to the characteristics of the application, by driving the input of data by the user, a HTML page is produced.

The input data can be submitted (see Figure 8) to the application according to three different modalities:

- by a data entry phase;
- by selecting a file resident on the client local disk;
- by choosing a link to a remote data source (a file located on a metacomputer machine).

In general, the application execution is expected to take a significant amount of time. Consequently, the user can leave its metacomputing session after submitting the application. At job completion the user is notified by an e-mail message. The message allows the user to reconnect and access the page of the results built by a proper Java servlet. Serious difficulties may arise when dealing with the transfer of large amounts of data due to the limited bandwidth available

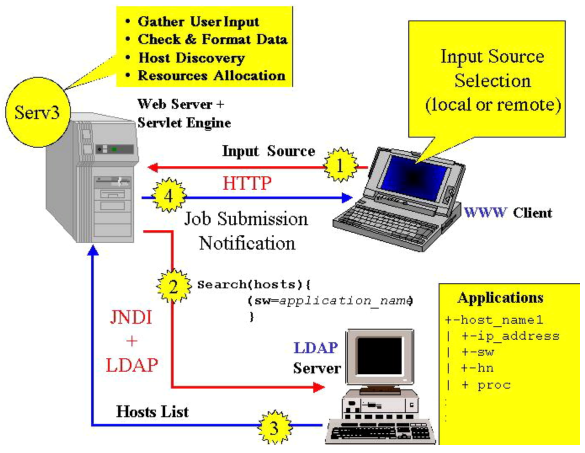

Figure8. Application Input Process.

on the network. Transfer time and data integrity cannot be guaranteed on the Internet.

3.2 The Middleware

The middleware layer consists of a Web and a LDAP servers. The Web server takes care of the interaction with the client and performs the Java servlets handling user's requests. The servlets residing on the Web server are:

serv1. Authentication of the user and his profile. According to the chosen policy, *serv1* handles the various phases of the authentication and the interaction with the client to establish his identity. To achieve this, the list of users allowed to make use of the services has to be accessed. This can be coded as an entry set of LDAP (see Figure 6). Among the attributes belonging to the user object there are those which identify the user profile (i.e. the applications he can submit for execution plus some auxiliary attributes useful for his identification). Auxiliary attributes can vary depending on the chosen policy. As an example, if an algorithm of the Challenge-Response type is implemented, the public key of the user should be stored (the assumption is that the decision on allowing access to the system should be hand made and left with the manager of the system).

These user profiles are necessary to state "who can do what" and "where should the results of a run be stored".

serv2. Application profiles management. According to the selected application, *serv2* needs to set the modality for transferring input data. To this end LDAP services are used too, in order to set input formats. In the DIT section of LDAP all entries related to a given application implemented in the back-end are defined. Its attributes define how data to be passed to executables have to be defined. As already singled out for the Client-end, there are three main ways of inputting data. According to the characteristics of the application *serv2* produces a HTML page that drives the user while inputting data (see Figure 8).

serv3. Validation of input data, resource localization, allocation and configuration, remote execution of the application, recollection and forwarding of results (see Figure 8 and 9). This is the most complex servlet which takes care of:

checking data format. Data input by the client need to match requirements set by the application. In case they do not, a HTML page is generated to inform the user about the error;

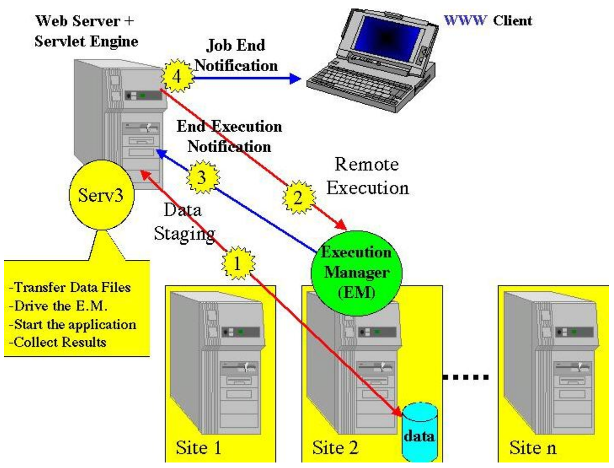

Figure9. Application Execution Process.

localizing resources. By interacting with the LDAP server it is figured out where the executable codes needed by the application are stored. It is worth noticing that LDAP is intrinsically static. Therefore, some mechanisms allowing a monitoring of the status of the resources of the back-end need to be introduced in order to allow also an update of the DIT entries guaranteeing the consistency of the information stored. This can be obtained by adding a further attribute to the object describing the characteristics of each machine. Aim of this attribute is to specify the date in which the last access to the resource has taken place. When the difference between the actual time and the time indicated by the attribute is larger than a predetermined amount, one can reasonably assume that the machine is available. Otherwise, the check is pushed to a lower level by using commands like *ping*, *top*, *procinfo* to update the entry related to the considered machine;

allocating and configuring resources previously localized. To this end a session is activated on the account of the user made available by

the the back-end machine using remote shell mechanisms. Input files are transferred into a given directory of the machine and a script to activate the executable codes of the application is configured;
remote executing of the application. A script is launched to start the execution of the application. System mechanisms like *pvm daemon*, *mpirun*, *Condor*, etc, local to the chosen machine, take care of configuring the virtual machine, of executing the parallel application and of storing the results on a file;
collecting and forwarding results. *serv3* waits until the application is ended before starting the collections of all its results. Then it opens a HTML page containing them or a link and forwards a mail to the user so that he can connect and access the desired information.

As already mentioned, the LDAP server, keeps the information about registered users, the characteristics of the available software and hardware. This information, initially provided by the systems administrator, due to the static nature of LDAP, is maintained by the *serv3* servlet that periodically updates the content of the LDAP entries according to the checks performed on the resources of the back-end. The Web and the LDAP servers interact via JNDI [16,17], an interface written in Java. This can be easily integrated into Java applications. The reason why JNDI has been chosen is that it has been developed with the aim of prividing access to a generic directory service and, at the present, it can interface not only LDAP but also NIS [16], DNS [18], and CORBA [19]. This guarantees to the applications that make use of it an easy extensibility.

3.3 The Back-end

The back-end is made by high performing computing resources, multiprocessor systems, workstation networks which provide computing power to the applications of the metacomputer. On these machines the Web server has user accounts that allow the execution of the applications.

4 Web-based Metacomputing Environments: Related Work

There is a growing number of worldwide projects related to metacomputing and grid computing [6,7]. Some of those focus on the exploitation of Java technology for Web-based metacomputing.

This section presents some of the most significative projects that are representative of the Web-based approach.

Charlotte [20], developed at New York University, was the first environment that has allowed any machine on the Web to participate in any ongoing computation. Charlotte is built on top of Java without relying on any native code.

Javelin [21] is a Java-based infrastructure for global computing. The system, developed at the Department of the University of California, Santa Barbara, is based on Internet and Web technology.

WebFlow [22], developed at the Northeast Parallel Architecture Center, is a computational extension of the Web model that can act as a framework for the wide-area distributed computing and metacomputing. The main goal of the WebFlow design was to build a seamless framework to publish and reuse computational modules on the Web so that end users, via a Web browser, can engage in composing distributed applications using WebFlow modules as visual components and editors as visual authoring tools.

NetSolve [23], developed at University of Tennessee and Oak Ridge National Laboratory, is a client/server application designed to solve computational science problems in a distributed environment. Netsolve clients can be written in C and Fortran, use Matlab or the Web to interact with the server. A Netsolve server can use any scientific package to provide its computational software.

Although our approach inherits some interesting solutions exploited in the previous mentioned projects, it is less general. In fact, our project focuses mainly on the creation of a Web-based metacomputing PSE to supply a completely transparent support to the user who does not have to care about the localization and the allocation of computing resources.

5 Conclusions

In this paper we have presented the main features of a PSE designed to facilitate the execution of a complex chemical application (SIMBEX) on a metacomputer through the Web. This project is developed in the framework of a European Communities COST Initiative-Action D23.

Our prototype is based on Web technologies and it is written in Java. The Java programming language successfully addresses several key issues related to grid environments. It also removes the need to install programs remotely; the minimum execution environment is a Java-enabled Web browser.

Many researchers agree with the fact that frameworks incorporating CORBA services will be very influential on the design of grid environments in the future. For this, we would like to investigate on the usage of CORBA technology to enhance some features of our PSE prototype.

6 Acknowledgments

We would like to thank the Master Thesis students, Fiorenzo D'Alberto and Andrea Vasapollo, who worked with us during the design and the development of this software environment.

References

1. S. Gallopoulos, E. Houstis, and J. Rice, *Computer as Thinker/Doer: Problem-Solving Environments for Computational Science*, IEEE Computational Science and Engineering, Summer (1994).

2. Metachem Workshop, European Community, Brussels, 26-27 November 1999.
3. C. Catlett, L. Smarr, *Metacomputing*, Communications of the ACM, 35(6), 44 (1992).
4. Baker M., Fox G., *Metacomputing: Harnessing Informal Supercomputers*, In High Performance Cluster Computing: Architectures and Systems, R. Buyya Ed., Volume 1, Prentice Hall PTR, NJ, USA (1999).
5. *The Grid: Blueprint for a Future Computing Infrastructure*, I. Foster and C. Kesselman Eds., Morgan Kaufmann Publishers, USA (1999).
6. W. Gentzsch (editor), *Special Issue on Metacomputing: From Workstation Clusters to Internet computing*, Future Generation Computer Systems, No. 15, North Holland, 1999.
7. M. Baker, R. Buyya, and D. Laforenza, *The Grid: International Efforts in Global Computing*, International Conference on Advances in Infrastructure for Electronic Business Science,and Education on the Internet (SSGRR'2000), L'Aquila, Italy, July 31 - August 6. 2000.
8. O. Gervasi, D. Cicoria, A. Laganà, and R. Baraglia Pixel 10, 19 (1994)
9. A. Patzer, *Introduction to Servlets, in Professional Java Programming*, Wrox Press Ltd (1999).
10. M. Wahl,T. Howes, and S. Kille, *Lightweight Directory Accesss Protocol*, RFC 2251, December (1997).
11. T.A. Howes, *The Lightwweight Directory Accesss Protocol:X.500 Lite*, Center for Information Technology Integration, July (1995).
12. A. Laganà and O. Gervasi,*A structured computational approach to chemical reactivity*, Chem. Phys., in press.
13. A. Laganà, G. O. de Aspuru, and E. Garcia, J. Chem. Phys. 108, 3886 (1998).
14. A.J.C. Varandas, *Multivalued Potential Energy Surfaces for Dynamics Studies*, A. Laganà and A. Riganelli Eds., in Lecture Notes in Chemistry, Springer-Verlag, in the press.
15. H. Krawczyk, M. Bellare,and R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, RFC 2104, February (1997).
16. M. Wilcox, *Server Programming with JNDI, in Professional Java Programming*, Wrox Press Ltd, December (1999).
17. JNDI - www.javasoft.com/products/jndi/index.html.
18. P. Mockapetris, *Domain Names - Concepts and Facilities*, RFC 1034, November (1987).
19. Object Management Group, Common Object Request Broker: Architecture and Specification, OMG Doc. No. 91.12.1 (1991).
20. A. Baratloo, M. Karaul, , Z.M. Kedem, and P. Wyckoff, *Charlotte: Metacomputing on the Web*, Special Issue on Metacomputing, Future Generation Computer Systems, pages 559-570, North Holland 1999. http://www.cs.nyu.edu/milan/charlotte/index.html
21. M.O. Neary, B.O.Christiansen, P.Cappello, K.E.Schauser *Javelin: Parallel computing on the Internet*, Special Issue on Metacomputing, Future Generation Computer Systems, pages 659-673, North Holland 1999. http://www.cs.ucsb.edu/
22. Haupt T., Akarsu E., and Fox G., Furmanski W, *Web Based Metacomputing*, Special Issue on Metacomputing, Future Generation Computer Systems, North Holland (1999) http://osprey7.npac.syr.edu:1998/iwt98/products/webflow/
23. H. Casanova and J. Dongarra, *NetSolve: A Network Server for Solving Computational Science Problems*, Intl. Journal of Supercomputing Applications and High Performance Computing, 11, 3, (1997). http://www.cs.utk.edu/ casanova/NetSolve/