Common Data Format for Program Sharing and Integration

Elda Rossi[1], Andrew Emerson[1], and Stefano Evangelisti[2]

[1]CINECA, via Magnanelli 6/3, 40033 Casalecchio di Reno (BO) – Italy
[2]Laboratoire de Physique Quantique, UMR 5626, Université Paul Sabatier, 118 Route de Narbonne, F-41062 Toulouse CEDEX – France

Abstract. This paper describes the design and implementation of a common data format within abiGrid, a grid-based project that connects different research groups of Quantum Chemists. The research activity of the partners is focused on orbital localization in a Multi-Reference context, since orbital localization is a necessary step towards the development of efficient methods for the treatment of large systems.The goal of abiGrid is to permit the use and interchange of home-made programs, while maintaining the individuality of the different codes that, as research tools, are subject to changes and evolution. Central points of the project are the design of a Common Data format and an extensive use of the Grid technology. In the present contribution the structure of the common data format is described. The format is based on XML, which has already used for chemical applications (CML). The problem of the large amount of data produced by ab-initio calculations is also addressed.

Keywords: Grid Technology, Meta-systems, Globus, XML-CML, ab-initio methods, Local Orbitals.

1 Introduction

The treatment of large systems is becoming a key issue in Quantum Chemistry (QC). Indeed it is hoped that in the near future it will be possible to apply QC techniques to fields involving macromolecules such as nano-sciences, biology, and so on. In this context, Linear-Scaling (LS) methods (algorithms whose computational complexity scales proportionally with the system size) are particularly interesting [1]. LS methods take advantage of the locality of most physical interactions to neglect distant contributions, thus achieving a linear growth. In order to use the locality of the interaction, however, it is essential to work with local orbitals, and this is the reason of a renewed interest on orbital-localization techniques in QC in the last years.

The applicability of local-orbital methods in a Multi-Reference (MR) context is presently studied by a group of researchers working on different aspects of Methodological and Applicative QC [2-8]. Since the different groups involved in the project are working on ab-initio codes developed in each single laboratory, program integration is a central point of this research, and a considerable effort is currently being devoted to this task. In order to work in this direction, two working group activities were proposed within COST in Chemistry [9].

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2658, pp. 316–323, 2003.

In the present contribution, we will focus on one of these activities, "AbiGrid: a meta-laboratory for code integration in ab-initio methods" [10], whose goal is to design and implement a tool, based on grid technology, that will enhance scientific cooperation by making easier the use and interchange of home-made programs, while maintaining the individuality of the different codes that, as research tools, are subjected to changes and evolutions dictated by the research activities of the different groups. All the partners involved in the Meta-Laboratory activity have been developing quantum chemistry codes for internal use for many years. These codes are complementary and their combined use is very important for new collaborations. On the other hand program sharing is difficult since the codes have not been written with sharing considerations in mind and no standard format has been respected in input and output planning.

The solution, in perspective, is the integration of all the codes into a single meta-system for QC calculations. This meta-system is not expected to substitute the general large packages for Quantum Chemistry available today on the market, like Gaussian or MolCAS or others, which are obviously much more complete and reliable. This is instead a collaborative environment for researchers that need to share their own home-made codes for common interests.

At the moment, we are focusing our work on the communication between the different codes. Two strategies can be adopted: one possibility is to write a series of one-to-one interfaces, connecting pairs of codes. The other possibility is to adopt a single data format, and to write interfaces from each code to this standard format. The first solution can be in some cases more desirable, particularly when two programs share already some data format. If the number of codes to be integrated becomes larger, however, the number of one-to-one interfaces becomes unreasonably large, and the use of a single data format is certainly preferable. Therefore, it was this solution that was adopted in our project.

2 The XML Choice

The strategy therefore was to decide on a common data format and write a series of "wrapper" codes in order to convert the input and output files from each application to or from the common data format.

The design of file formats able to represent particular types of information is a common problem but is particularly pertinent to the area of computational chemistry. In general, no standard format has ever been adopted for any area of chemistry with most software providers tending to use their own proprietary solutions. However, in recent years developments in the evolution of the creation of documents destined for the WEB has led to a new formalism termed XML (eXstensible Mark-up Language) [11]. XML is not a new computer language but rather a "meta-language", a set of rules which can be used to generate a syntax and vocabulary appropriate to the data to be described. In this way many "flavors" of XML have been created to represent information from many diverse areas (e.g business, medicine, archaeology, etc.) as well as the successor to HTML for web-page design. The main advantage of an XML-based data format is that it is extensible, i.e. new types of data can be easily and seamlessly added to the language definition if the need arises – something which is not possible with the currently available file types. In addition, since XML documents

contain just freely formatted text they are human readable and can be easily interpreted by text parsers and other programs and converting between them provides a lossless mechanism of data transfer. There are also many readily available utilities for processing XML documents which can be transferred over the internet using existing and commonly used protocols (e.g. hypertext transfer protocol or http).

To our knowledge there is only one mature implementation of an XML for chemistry; this has been named Chemical Mark-up Language or CML and has been designed by Peter Murray-Rust and Henry Rzepa [12]. There has yet to be a widespread adoption of CML by the computational chemistry community but interest is growing and the need for a more unified approach to storing chemical information is generally accepted. Although in principle any kind of data can be included in CML, it has been primarily designed for representing molecular structures and chemical reactions rather than for QC quantities. The authors of CML are also in the process of formulating an XML definition called CCML, or Computational Chemistry Mark-up language, which is expected to be able represent information relevant to computer simulation such as algorithms, parameterizations (e.g. force-fields) and job control. However, neither CML nor the forthcoming CCML was designed with quantum chemistry properties in mind so we have therefore decided to construct our own XML definition (or "XML schema") appropriate for handling data in QC calculations. We should point out that having our own XML (tentatively designated QCML, for Quantum Chemistry Mark-Up Language), does not imply abandoning CML or CCML completely because XML allows one to have multiple "namespaces"

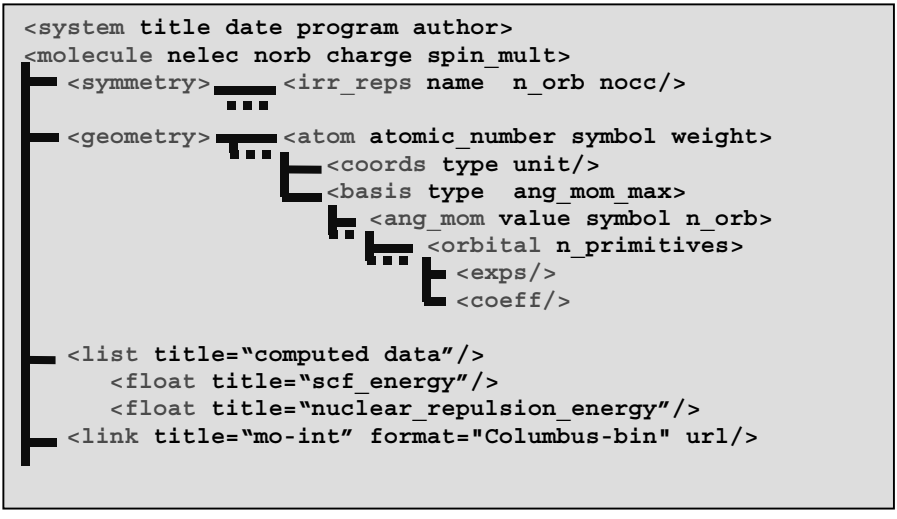

Fig. 1. A preliminary schema of QCML. A QC computation can be described with a limited number of "tags" and "attributes"

(i.e. mark-up language definitions) within the same document. Thus, although we expect to use mainly QCML in our files, we are free to use CML if it is more appropriate for the data item in question (e.g. for molecular structures). The design of

QCML is still at an early stage but there will be tags such as <symmetry>, <geometry>, <coefficient>, etc together with appropriate attributes for describing the quantities required or outputted by the QC applications. Eventually a schema will be published which will allow the validation of any QCML document. In fig. 1 a preliminary schema of QCML is shown.

We point out in particular the use of the general CML "list/float" tag to maintain computed quantities like different types of energy and the "link" tag that will be discussed in another section (XML and large data files).

3 XML Wrappers and Tools

All the data describing the chemical system of interest are kept in a "data repository" as showed in fig. 2. The data repository is based on the XML format and is somewhere on the network.

All the efforts for integration and conversion from/to the common format are handled by "wrappers", programs specifically designed for each single code that generate the input files starting from the repository (INPUT wrapper) and that, at the end, store important data from the output files into the repository (OUTPUT wrapper).

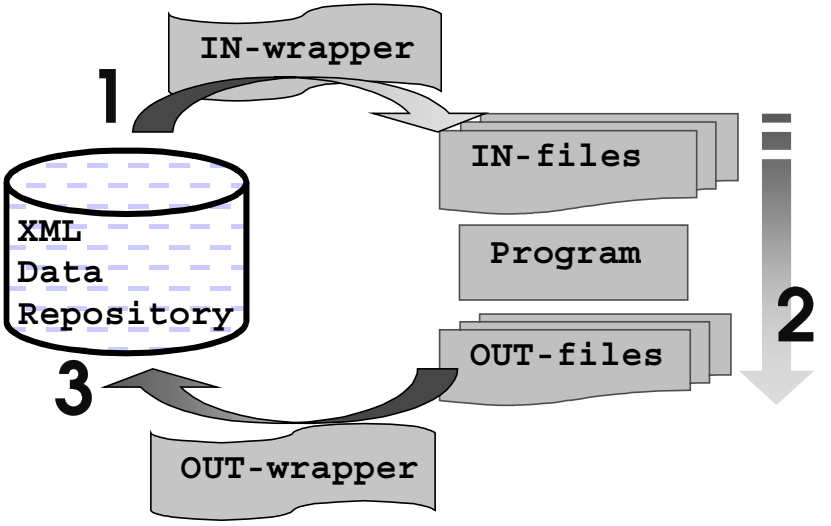

Fig. 2. The input wrapper converts data from the repository to the input files (1), the code runs as usual producing output files (2),at the end the output wrapper converts selected output data to the repository (3)

This strategy allows us to leave the program unchanged. If we modify the program or we add a new one, we only have to modify or to write the specific wrappers. So in theory the meta-system will not be affected by the change in any other way.One important task therefore is to write the wrapper programs which will perform the conversions to and from this common format which could also be used for documentation purposes and with appropriate utilities for results analysis. To do this we have adopted two strategies:

1. The writing of a number of PERL scripts to perform the conversions between the application files and the QCML data format.
2. The construction of a FORTRAN library of low-level and high-level subroutines designed to aid the reading and writing of XML documents.

PERL was chosen because it is one of the most powerful text processing languages and there is already an extensive library of modules for parsing, writing and validating XML documents [13]. In addition, PERL interpreters are generally found on all UNIX systems and since it does not need to be explicitly compiled, transferring the PERL wrappers between the different hardware and software architectures in a grid environment is straightforward.

The FORTRAN library was written at the request of the application program authors and maintainers to facilitate the integration of their codes into the meta-system. FORTRAN in fact is well known and widely used throughout this community.

Although FORTRAN is not usually recommended for text manipulation, it is possible (particularly with FORTRAN 90) and the routines can be validated with their PERL equivalents.

The routines in the library are organised in three levels: the top level doing macro tasks, the intermediate levels that are used by the previous ones but can also be used by the user and a low level that contains routines mainly of internal use.

In fig.3 an example of a top level routines is presented. The routine **`Search_tag`** searches the repository (trough the **`file`** argument) for a given tag (**`atom`** in the example). It is possible to specify what atom (**`id3`**, that is the atom no. 3, in the example) and what specific attribute we are interested in (**`symbol`**). The result will be the symbol of that atom (**`Ca`** for example) stored as a string in the **`value`** argument.

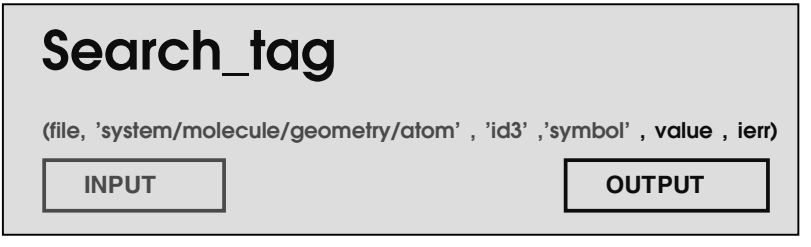

Fig. 3. The **`search_tag`** routine from the FORTRAN library

4 XML and Large Data Files

XML, as well as being text-based, is also rather verbose and requires more disk space than other data formats in order to store the equivalent quantity of data. This is not usually considered important bearing in mind the storage available on current computer systems but QC codes very often produce and access huge quantities of data, such as electron integrals or checkpoint files. For efficiency, both in terms of disk space and CPU time, these files are usually written in a binary format.

Due to the size of the stored data (perhaps even hundreds of Gbytes for very large calculations), it is unthinkable to keep these data in a formatted XML file. Although large XML files can be compressed quite efficiently with standard compression utilities at the end of a run, the increase in data access time during program execution would severely affect program performance. Thus, when large binary files are required it was decided to create an XML file containing a link to the location of the binary file, together with information describing its format and contents. Although not ideal there seems to be no other workaround without affecting program performance.

5 The Implementation

A number of existing programs have been used as building blocks for the prototypal meta-system. In most cases the codes are home-made, but sometimes they are general use packages, distributed freely over the internet (as is the case for Dalton and Columbus). The programs in question are as follows:

- COLUMBUS (General ab-initio electronic package) [14]
- DALTON (General ab-initio package) [15]
- CAS-DI (Multi-Reference Configuration Interaction) [16]
- EPCISO (Spin-orbit Configuration Interaction) [17]
- NEVPT (Multi-Reference Perturbation Theory, Perturbative-Variational approaches) [18]
- LOCNAT (Localized Multireference algorithm) [3,4]
- FCI (Full Configuration Interaction): [19]
- PROP (Property Calculation) [20]

Some of these codes can be considered “zero level” programs, as they do not require pre-computed structured data. On the other hand, a program like FCI needs molecular orbital integrals to start. It has been designed to take those data from a specific commercial program, but allowing the integrals to come from other sources will increase flexibility.

Another problem we are facing is that of “multiple instances” of the same input data. For example, all codes need information like “number of atoms”, “electrons”, “basis set”, etc., each of them in a different input format. The solution is to collect all data at the beginning and transfer them to each program in a suitable format.

The first prototype of this implementation, was the interface between the COLUMBUS and FCI chains. This was because the FCI code requires a very limited

set of data (essentially, one- and two-electron integrals), and contains a rather limited set of options. However, we notice that the strategy adopted will permit a future integration of the other codes within the same XML scheme. Work has already started on the PERL parsers needed to convert the various input and output files to their QCML equivalents. The main problem being encountered is not so much the writing or parsing of the XML code, the library routines make this a simple programming task, but instead gathering together all the quantities present in the many files required and generated by the applications.

6 Conclusion

We are well on the way to interfacing different ab-initio codes via an XML wrapper, which permits the data transfer from one code to the other. The problem of the large amount of data produced and/or needed by most of the QC codes (Orbital Coefficients, Molecular Integrals, Configuration Coefficients, etc) and for which a verbose solution like XML does not seem appropriate, was solved by writing in the XML file the locations and descriptions of the binary files containing the data. We are working on the extension of this approach to the different codes involved in the abiGrid project. We believe that the proposed solution, in which the different codes maintain their individuality, and are under the direct responsibility of the groups which wrote them, can represent an efficient way to enhance the collaboration between different researchers belonging to the Quantum Chemistry community.

References

[1.] S. Goedecker, Rev. Mod. Phys., 71 (1999) 1085
[2.] N. Guihéry, J.-P. Malrieu, S. Evangelisti, and D. Maynau, Chem. Phys. Lett. 349 (2001) 555
[3.] D. Maynau, S. Evangelisti, N. Guihéry, C.J. Calzado and J.-P. Malrieu, J. Chem. Phys. 116 (2002) 10060
[4.] C. Angeli, S. Evangelisti, R. Cimiraglia and D. Maynau, J. Chem. Phys. 117 (2002) 10525
[5.] C. Angeli, C.J. Calzado, R. Cimiraglia, S. Evangelisti, N. Guiéhry, J.-P. Malrieu, and D. Maynau, J. Comp. Meth. Sci. and Engen. 3 (2002) 1.
[6.] C.J. Calzado, S. Evangelisti, D. Maynau, J. Mol. Struct. (THEOCHEM), in press.
[7.] C. Angeli, C.J. Calzado, R. Cimiraglia, S. Evangelisti, N. Guihéry, T. Leininger, J.-P. Malrieu, D. Maynau, J.V. Pitarch, and M. Sparta, Mol. Phys. in press.
[8.] C. Angeli, C.J. Calzado, R. Cimiraglia, S. Evangelisti, and D. Maynau, Mol. Phys. in press.
[9.] http://cost.cordis.lu/src/home.cfm
[10.] http://cost.cordis.lu/src/extranet/publish/D23WGP/d23-0006-01.htm
[11.] see for example http://www.w3.org/XML
[12.] Murray-Rust, Henry S. Rzepa and Michael Wright, New J. Chem., 2001, 618–634. (http://www.xml-cml.org/
[13.] See, for example, the CPAN archive, http://www.cpan.org
[14.] http://www.itc.univie.ac.at/~hans/Columbus/columbus.html; H. Lischka, R. Shepard, R. M. Pitzer, I. Shavitt, M. Dallos, Th. Müller, P. G. Szalay, M. Seth, G. S. Kedziora, S. Yabushita and Z. Zhang, Phys. Chem. Chem. Phys. 3 (2001) 664.

[15.] http://www.kjemi.uio.no/software/dalton/dalton.html
[16.] N.Ben Amor, D. Maynau Chem. Phys. Lett. 286 (1998) 211.
[17.] V. Vallet, L. Maron, C. Teichteil et J.P. Flament, J. Chem. Phys. 113 (2000) 1391.
[18.] C.Angeli, R.Cimiraglia, S.Evangelisti, T,Leininger, J.-P. Malrieu, J.Chem. Phys. 114 (2001) 10252.
[19.] G.L. Bendazzoli, S.Evangelisti, J.Chem. Phys. 98 (1993) 31.
[20.] J. Pitarch-Ruiz, J. Sánchez-Marín, D. Maynau. J. Chem. Phys. 112 (2000) 1655.