PROCEEDINGS OF SCIENCE

High Performance Grid Computing: getting HPC and HTC all together

A. Laganà*
Department of Chemistry, University of Perugia, Perugia, Italy (IT)
E-mail: lag@unipg.it

C. Manuali
Department of Chemistry, University of Perugia, Perugia, Italy (IT)
E-mail: carlo@unipg.it

S. Rampino
Department of Chemistry, University of Perugia, Perugia, Italy (IT)
E-mail: ser_ram@dyn.unipg.it

A. Costantini
INFN-CNAF - National Institute of Nuclear Physics, Bologna, Italy (IT)
IGI - Italian Grid Infrastructure, Italy (IT)
Department of Mathematics and Informatics, University of Perugia, Perugia, Italy (IT)
E-mail: alessandro.costantini@cnaf.infn.it

E. Rossi
CINECA, Casalecchio di Reno, Italy (IT)
E-mail: e.rossi@cineca.it

M. Carpené
CINECA, Casalecchio di Reno, Italy (IT)
E-mail: m.carpen@cineca.it

A. Ghiselli
INFN-CNAF - National Institute of Nuclear Physics, Bologna, Italy (IT)
E-mail: antonia.ghiselli@cnaf.infn.it

M. Cecchi
INFN-CNAF - National Institute of Nuclear Physics, Bologna, Italy (IT)
E-mail: marco.cecchi@cnaf.infn.it

PoS(EGICF12-EMITC2)144

PROCEEDINGS OF SCIENCE

The recently developed Grid Framework, GriF, is a Workflow Management System designed to evaluate Service and User Quality when helping the Grid users (especially those of the COMPCHEM Virtual Organization) in optimizing the return of the parameter sweeping studies. In this perspective, GriF has now been extended to allow access to High Performance supercomputers from the Grid. The implementation of such High Performance Grid has requested a tight collaboration of various experts of the Italian Grid Initiative to the end of bridging the gLite middleware with that of some supercomputer platforms. The advantages achievable in this way are illustrated by discussing the use case of atom-diatom quantum reactive scattering calculations.

PoS(EGICF12-EMITC2)144

EGI Community Forum 2012
EMI Second Technical Conference
26-30 March, 2012, Munich, Germany (DE)

*Speaker.

1. Introduction

The main difficulty for the generic users of the Grid (as most of the members of the virtual communities are) is the high level of computer skills still required to run distributed massive applications on it. To this end, middleware consortia (like EMI [1]) and some specific projects of the European Grid Initiative EGI [2] (like EGI-Inspire [3], SCI-BUS [4] and MAPPER [5]) are working to develop user friendly tools which are aimed at managing Grid operations (as for example running jobs, checking their status and retrieving related results) with no need for mastering the low-level environment.

Among these tools is GriF [6, 7]. GriF is a Workflow-oriented Grid Framework designed within the activities of the Virtual Organization (VO) COMPCHEM [8] having the specific purpose of exploiting the evaluation metrics adopted by the VO to optimize its organization and services [9]. To this end, GriF has been structured so as not to require the users to bear Grid Certificates (by adopting Robot Certificates [10]), to handle various types of programs including the binary ones and to choose as well which Grid resources are to be used on the basis of specific Quality of Service (QoS) and Quality of User (QoU) evaluation procedures [9]. Another specific feature of GriF is to allow user operations relying on both simple commands and various Framework-side interventions offered as Grid Services. All this is addressed in the present paper to the discussion of the optimization of the selection of the machines of the Grid platform with respect to their availability and to their suitability for large applications by taking as a benchmark quantum reactive scattering calculations of the Molecular and Materials Science and Technology (MMST) field [11, 12].

For the same purpose, in the present paper also the challenging target of assembling highly complex Workflows (including the so called Workflow of Workflows) has been considered to the end of discussing how to provide the users with the opportunity of pushing multiple scientific applications to an unprecedented level of integration by executing sequential, conditional and/or iterative code paths.

In practice, thanks to the convergence of interests of the Italian Grid Initiative (IGI) [13] and of the supercomputer centre CINECA [14], our work has focused on enabling GriF to selectively route computational tasks from Grid to supercomputer machines when dealing on the Grid with highly realistic complex collaborative (in terms of expertise and platforms in particular) scientific applications which, depending on the size of related calculations, may shift from the use of ordinary Grid nodes to those of supercomputers.

Accordingly, the paper is articulated as follows: in section 2 the structure of the simulator and the general quantum frame of MMST applications are discussed; in section 3 the main features offered by the GriF flowchart based on two Java servers and a Java client, are illustrated; in section 4 the procedure using the Workflow-enabled version of GriF to submit computational tasks from the Grid to outside supercomputers, is described; in section 5 performance results and an enhancement of the parametrization of the application to deal with multiple Workflows are presented. Some conclusions are outlined in section 6.

2. GEMS: a mixed High Throughput - High Performance application

As already mentioned, the design of GriF was prompted by the need of providing the mem-

bers of the COMPCHEM VO with a tool for optimizing the use of complex simulators like the so called Grid Empowered Molecular Simulator (GEMS) [15, 16] that calculates in an ab initio fashion MMST properties [17]. The foundations of GEMS were first discussed in ref. [18] and later implemented in a portal developed for the a priori simulation of crossed beam experiments named SIMBEX [19]. The present flow-chart of GEMS is definitely more general and structured than the one adopted by SIMBEX and may require both High Throughput (HTC) and High Performance (HPC) Computing capabilities. It consists, as shown by the schematic representation given in Fig. 1, of the conditional execution of four blocks which target, respectively, the following tasks:

1. INTERACTION: Evaluate, if not otherwise available, the electronic structure of the molecular aggregate of interest (*BLOCK 1*);
2. FITTING: Fit, if needed, the calculated ab initio values to generate a Potential Energy Surface (PES) using an appropriate functional form (*BLOCK 2*);
3. DYNAMICS: Integrate on an available or generated PES the set of scattering equations most suited for describing the motion of the nuclei of the considered system (*BLOCK 3*);
4. OBSERVABLES: Average over multiple valued parameters the scattering quantities needed to work out a theoretical estimate of the requested observable property (*BLOCK 4*).

BLOCK 1: The distribution of the INTERACTION computational tasks starts by assigning single molecular geometry ab initio calculations to each computing unit if related values are not already available. Every ab initio computational task determines (for a given molecular geometry corresponding to a fixed nuclei position vector $\mathbf{R}$) the corresponding set $E_e(\mathbf{R})$ of electronic energy eigenvalues of the stationary Schrödinger equation for the electronic Hamiltonian $\hat{H}_e$:

$$\hat{H}_e\Phi_e(\mathbf{r};\mathbf{R}) = E_e(\mathbf{R})\Phi_e(\mathbf{r};\mathbf{R}) \tag{2.1}$$

(in which $\hat{H}_e = \hat{T}_e + V_{eN} + V_{ee}$ with $\hat{T}_e$ being the electron kinetic operator, V_{eN} the electron-nuclear attractive Coulomb potential and V_{ee} the electron-electron repulsive Coulomb potential) worked out within the Born-Oppenheimer approximation by assuming that the system wavefunction $\Phi_e(\mathbf{r};\mathbf{R})$ can be parametrically separated into an electronic and a nuclear component. Such equation is usually integrated by expanding for each fixed molecular geometry $\Phi_e(\mathbf{r};\mathbf{R})$ into a suitably large basis set of appropriate functions $\{\varphi(\mathbf{r})\}$ whose expansion coefficients are the eigenvectors $\mathbf{c}_I(\mathbf{R})$ [20] for each electronic eigenstate I. The associated computational tasks can be executed on the ordinary nodes of the Grid only at low level of theory. When high level of theory electronic eigenvalues are seeked, supercomputer facilities need to be used so as to rely on large virtual memories suited to support huge matrices and extended diagonalization procedures [21]. For this type of calculations a large variety of well established (and well described in specific references) packages are already considered by our GEMS implementations for atom-diatom systems such as the GAMESS-US [22] and the DALTON [23] codes indicated in Fig. 1.

BLOCK 2: No distribution is usually considered for the FITTING block associated with the generation of an analytical formulation of the PES out of the pointwise representation of the interaction produced by the previous block or downloaded from the Internet network. The related procedure (usually of the least square type) adopts a functional representation of the electronic energies easy

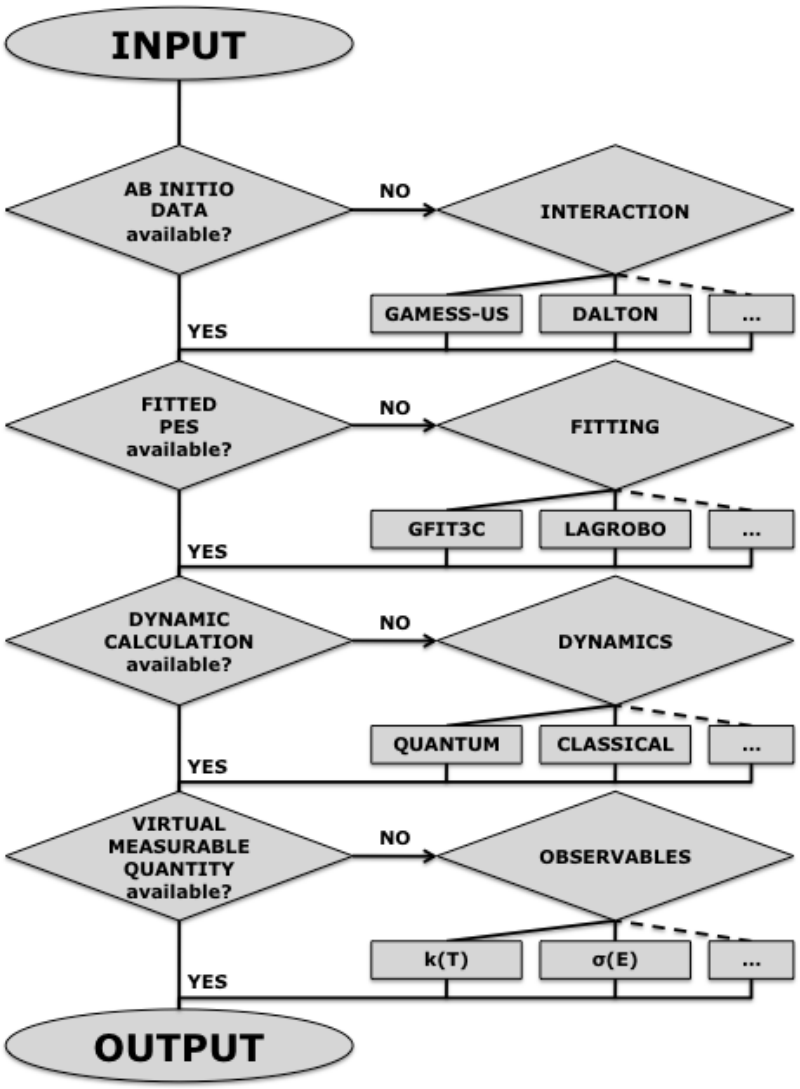

Figure 1: The simplified flow-chart of GEMS.

to modify and simple to calculate (see for example ref. [24]). Already considered in GEMS are the three atom routines GFIT3C [25] and LAGROBO [26] indicated in Fig. 1 which usually run also on the ordinary nodes of the Grid.

BLOCK 3: The most elaborated distribution schemes were adopted for the packages of the DYNAMICS block which tackle the problem of integrating the electronically adiabatic (in the *Ith* electronic state) Schrödinger equation in its Time Dependent (TD) form:

$$i\hbar\frac{d}{dt}\Psi_{NI}(\mathbf{R},t) = [\hat{T}_N(\mathbf{R}) + V_I(\mathbf{R})]\Psi_{NI}(\mathbf{R},t) = \hat{H}_{NI}(\mathbf{R})\Psi_{NI}(\mathbf{R},t) \tag{2.2}$$

(in which $\Psi_{NI}(\mathbf{R},t)$ is the TD nuclei wavefunction and $\hat{T}_{NI}(\mathbf{R})$ is the (time independent) nuclei kinetic operator of the electronic state I while $V_I(\mathbf{R})$ is the potential energy function providing a proper analytical representation of the ensemble of the needed $E_I(\mathbf{R}) + V_{NN}$ values with V_{NN}

being the nuclear-nuclear repulsive Coulomb potential. As an alternative the Time Independent (TI) formulation:

$$\hat{H}_{NI}(\mathbf{R})\Xi_{NI}(\mathbf{R}) = [\hat{T}_N(\mathbf{R}) + V_I(\mathbf{R})]\Xi_{NI}(\mathbf{R}) = E\Xi_{NI}(\mathbf{R}) \tag{2.3}$$

(in which $\Xi_{NI}(\mathbf{R})$ is the stationary nuclei wavefunction ($\Psi_{NI}(\mathbf{R},t)$=$\Xi_{NI}(\mathbf{R})e^{-iEt/\hbar}$) and E is the (constant) total energy of the considered molecular system [27]) can be adopted. A variety of methods have been implemented in GEMS for this purpose (mainly developed in-house or adapted from scientific libraries) though still restricted to atom-diatom systems. These reactive scattering codes are both of the TD (RWAVEPR) [28] and the TI (ABC) [29] type. They either expand the TI nuclei wavefunction in a suitable basis set parametrically dependent on a molecular geometry coordinate (used as a continuity variable) expressing the advance of the reactive process (reaction coordinate) or collocate the TD wave packet on a proper coordinate grid and use time as a continuity variable. However, as we shall discuss in more detail in section 4 even for atom-diatom reactions (the simplest reactive system), related calculations only seldom can be re-conducted to simple parameter studies applications in which a large quantity of independent tasks are executed. The involved differential equations, in fact, are so demanding in terms of memory (because of the size of the used matrices) that, when increasing the complexity of the interaction or even just seeking convergency with the total angular momentum quantum number J, the computational tasks can not be handled by the ordinary nodes of the Grid.

Yet, for large systems the completely alternative VENUS [30], GROMACS [31] and DLPOLY [32] typical packages (also implemented in GEMS) based on the integration of a set of classical mechanics differential equations starting from different sets of initial conditions are used. For moderately large systems these calculations are, again, typical parameter studies applications in which a large quantity of independent tasks, each iterating over time the step-wise integration of the related classical mechanics coupled differential equations, can be easily dealt by the ordinary nodes of the Grid because requiring a limited amount of memory and an affordable (for the Grid) demand of computing time.

BLOCK 4: No distribution is usually considered also for the OBSERVABLES block of the present version of GEMS that has been specialized for the simulation of the product beam intensity of crossed beam experiments [33, 34]. This block is, however, strictly experiment dependent and, therefore, most often not even distributed on the Grid.

To enable an appropriate distribution of the computational tasks of GEMS we have in the past performed several studies and reported related results in the literature either by using in-house developed scripts or by further developing the already mentioned framework GriF [35]. We found that GriF improves not only on the in-house developed scripts (because of its generality) but also on P-GRADE [36] (because of its higher efficiency in selecting resources and carrying out operations based on friendly User Driven Services). Both aspects meet, as already mentioned, the goals of the COMPCHEM users because they allow an easy integration, generation and control of the activities of the various computational tasks by managing the flow of the application and determining what and how data is captured and used. In particular, GriF provides the users not only with the "static" possibility of deciding once for ever the packages to be composed but also with the "dynamic" possibility of building run-time options like the insertion of barriers to synchronize computing paths, the implementation of iterations (including the condition controlled ones) to check self consis-

tency and convergence of the calculated properties as well as the preferential routing of the tasks to specific platforms [37, 38].

3. The Structure of the GriF Framework

Thanks to the just mentioned characteristics, GriF is able to lead to an optimal usage of the memory, to a reduced engagement of the cpu, to a minimal consumption of wall time and to an optimized distribution of the tasks over the network allowing so far the assemblage of applications of higher complexity and the exploitation of collaboration among experts of various (complementary) research areas.
As already mentioned, in fact, GriF is able to capture, out of the data supplied by the Grid monitoring sensors, the information relevant to work out the QoS parameters useful to select the computational resources suited for structuring complex applications under the form of optimized Web Services [39] implementing a Service Oriented Architecture (SOA). The SOA organization of GriF, as shown by the flow-chart given in Fig. 2, consists essentially of two Java servers and one Java client [40].

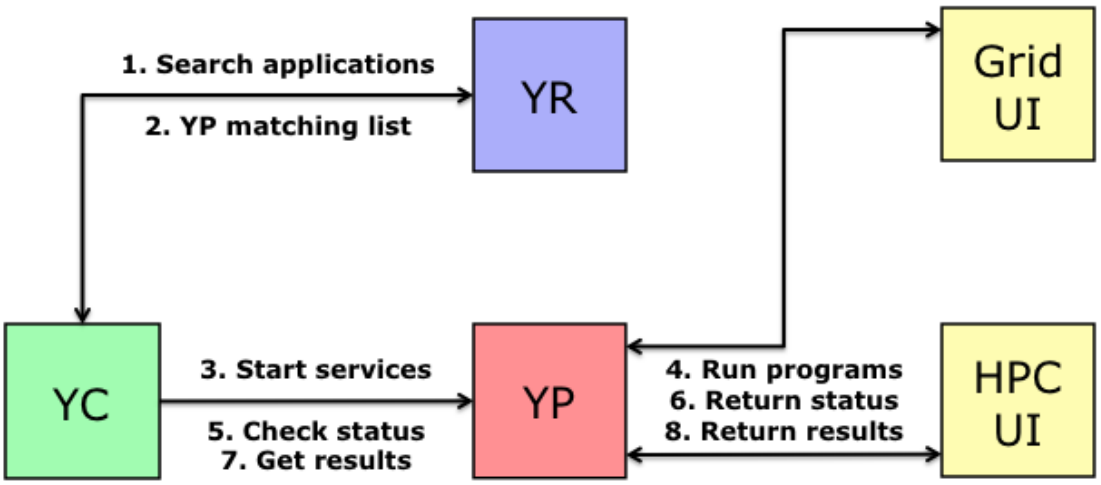

Figure 2: The simplified flow-chart of GriF with YC being the Consumer, YR being the Registry, YP being the Provider and UIs being the entry-points to the computational platforms.

The two Java servers, called YR (acting as an information registry for the available services) and YP (acting as a container providing Grid Services), both illustrate services and reference self-describing interfaces which are published in platform-independent documents. The Java client YC takes care of running the jobs on the associated User Interface (UI) and supports the correct interfacing of the Grid Services offered by GriF.
To run a program (or a set of programs) on the Grid the user can utilize YC to perform several transparent actions devoted to:

- The management of the basic Grid operations (e.g. checking single and multiple job status on any kind of devices);
- The search of the programs on YR (e.g. searching for a YP exposing a Web Service offering the execution of the requested programs);
- The optional introduction of changes in the requested programs;
- The compilation of new executables on the selected YP by replacing the ones provided as default.

Typical fragments of simplified code used respectively in YP (to wrap Grid applications into Grid Services) and in YC (to invoke the Grid Services exposed by the YP), are listed in Fig. 3. In the YC section of Fig. 3, the variable `YP_NAME` takes the value of the YP name selected by the user during the Service Discovery phase performed using YR (see Fig. 2). Then a Grid Service is invoked on YP waiting for the results (see last line of the YC section). In the YP section, after a method called `run` has received the date, the user name and the specific operation input data as input, a shell script called `rungrid.sh` is invoked in order to perform the actual low-level Grid operation. Accordingly, then related results are returned back to the Grid Service and then to YC.
Users can also utilize the results of any previous actions to start the execution of the Grid job after passing a user-specific input file (when the default one is not appropriate), monitor the jobs status and eventually retrieve the results.
All these features of GriF were utilized from the very beginning to wrap user programs, manage the Grid Certificates and Proxies, compile different executables, execute Grid jobs, monitor the jobs status and retrieve the results so as to make the services easy-to-use. As a matter of fact, using GriF the members of a Grid Community (COMPCHEM, for example) are able to set up and manage the YPs to expose their services in an open, standard and secure way with no need for them to be familiar with the wrapped programs and the Grid platform.
In other words, GriF provides the members of a Grid community with a black-box like user friendly tool enabling the exploitation of the innovative features of Grid computing by making unnecessary to master related technicalities. At the same time, GriF offers to the users services of increasing level of complexity. Thanks to its robust SOA framework nature GriF provides, in fact, support to collaboration among researchers bearing complementary expertise and collaborating in multi task computational applications.
A key problem of building complex applications (like those targeted by GEMS) is, indeed, the need for combining codes engaging computing resources in a variable and uneven fashion. To the end of distributing on the Grid massive concurrent executions, the wrapped services of GriF were enabled to make use of the Collection option [41] typical of the gLite (Lightweight Middleware for Grid Computing) middleware [42].
As a result, after choosing between a custom binary program or an application already available under the form of Grid Service, the user has to specify, in general, a compressed file in which all the different inputs (one for each subjob to be run concurrently) are stored. To this end, GriF makes use of the Simple Object Access Protocol (SOAP) with Attachments (SwA) [43] XML-based messaging protocol to the end of allowing file transfers between YC and YP and vice-versa. To provide more detail of the work done we describe here (see Fig. 4 and 5, respectively) part of

YP Web Service code:

```
public class RunGrid {
   public String run(String input, String user, String date) {
      String cmd = ''rungrid.sh'' + input + user + date + ''>'' + outputfile;
      String[ ] command = {''sh'', ''-c'', cmd};
      Process p = null;
      Runtime r = Runtime.getRuntime();
      p = r.exec(command);
      p.waitFor();
      FileReader file = new FileReader(outputfile);
      BufferedReader reader = new BufferedReader(file);
      while ((line = reader.readLine()) != null) output += line;
      return output;
   }
}
```

YC Client code:

```
public class YC {
   public static void main(String [ ] args) {
      String endpoint = ''http://'' + YP_NAME + ''/RunGrid.jws'';
      Service service = new Service();
      Call call = (Call) service.createCall();
      call.setTargetEndpointAddress(new java.net.URL(endpoint));
      call.setOperationName(''run'');
      call.addParameter(''input'', XMLType.XSD_STRING, ParameterMode.IN);
      call.addParameter(''user'', XMLType.XSD_STRING, ParameterMode.IN);
      call.addParameter(''date'', XMLType.XSD_STRING, ParameterMode.IN);
      call.setReturnType(XMLType.XSD_STRING);
      String result = (String) call.invoke(new Object [ ] {input, user, date});
   }
}
```

Figure 3: Part of the (simplified) code implementing the Web and Grid Services wrapping and invoking.

the (simplified) codes of the sections implementing the upload of an application from YC to YP and retrieving of the results from YP to YC.

Fig. 4 shows (in bold) the `DataHandler` Java class type reference for the sending of the the application file from YC (line 14 of the YC code section) as well as the `String` for the receiving of the related file by YP (line 3 of the YP code section). Fig. 5 shows the definition of the timeout (unlimited to allow the handling of very large files) when YC gets a results' file (line 4 of the YC code section) and the request to YP for it (called `filename`) by making use of the `DataHandler`

YC code (*send*):

```
File f = fileUploadChooser.getSelectedFile();
Service service = new Service();
try {
    QName qnameAttachment = new QName("xsd:Binary");
    // When the attachment is in ASCII format use the following line:
    // QName qnameAttachment = new QName("xsd:ASCII");
    Call call = (Call) service.createCall();
    call.setTargetEndpointAddress(new java.net.URL(endpoint_upload));
    call.setOperationName("upload");
    call.registerTypeMapping(DataHandler.class, qnameAttachment, \\
        JAFDataHandlerSerializerFactory.class, \\
        JAFDataHandlerDeserializerFactory.class);
    call.setReturnType(XMLType.XSD_STRING);
    DataHandler dh = new DataHandler(new FileDataSource(f));
    call.addParameter("dh", qnameAttachment, ParameterMode.IN);
    String exit = (String) call.invoke(new Object[]{dh});
}
```

YP code (*receive*):

```
public class Upload {
    public String upload(DataHandler dh) {
        String attach = dh.getName();
        [...]
        return "ok";
    }
}
```

Figure 4: Part of the (simplified) code implementing SwA from YC to YP.

type mentioned above (line 12 of the YC code section). The figure shows also the response of YP and the returning of the requested file by making use of the `FileDataSource` Java class type (line 3 of the YP code section).

Fig. 6 (where two typical SOAP messages, formed respectively by the Envelope and the related Body with at least an element, are shown) illustrates, instead, the correlation between the elements of a typical SOAP message (where the 'In message' section is the SOAP request and the 'Out message' section is the SOAP response) as well as some variables and values used by YC and YP. It is worth noticing here (line 12 of the 'In message' section) the use of a CID (Content-ID, the scheme providing identifiers for messages and their body parts [44]) URL (see the `href` attribute) by which the attachment is associated to the `dh` (DataHandler) parameter.

Therefore, after the job has started and each related subjob is run on the Grid, all the relevant information, like for example the URL where to check the job's status, are returned to YC. In this respect, for each subjob belonging to the same job collection, a different Computing Element (CE) queue (q) is used (and also returned to the user) according to the so called 'Ranking' feature of GriF (the ability of evaluating the Quality of each q by making use of some in-house developed adaptive algorithms [45]).

YC code (*request*):

```
File f = fileGetChooser.getSelectedFile();
Service service = new Service();
try {
    ((org.apache.axis.client.Call)call).setTimeout(new Integer(0));
    QName qnameAttachment = new QName("Result", "DataHandler");
    Call call = (Call) service.createCall();
    call.setTargetEndpointAddress(new java.net.URL(endpoint_get));
    call.setOperationName(new QName("Result", "get"));
    call.registerTypeMapping(DataHandler.class, qnameAttachment, \\
        JAFDataHandlerSerializerFactory.class, \\
        JAFDataHandlerDeserializerFactory.class);
    call.setReturnType(qnameAttachment);
    call.addParameter("filename", XMLType.XSD_STRING, ParameterMode.IN);
    DataHandler ret = (DataHandler) call.invoke( new Object [] {filename});
    FileOutputStream fos = new FileOutputStream(f);
    ret.writeTo(fos);
    fos.close();
}
```

YP code (*response*):

```
public class Get {
    public DataHandler get(String filename) {
        DataHandler dh = new DataHandler(new FileDataSource(filename));
        return dh;
    }
}
```

Figure 5: Part of the (simplified) code implementing SwA from YP to YC.

```
=======================================================
= Elapsed: 1525 milliseconds

= In message:
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <upload soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <date xsi:type="xsd:string">20101004083459</date>
            <WSname xsi:type="xsd:string">Upload</WSname>
            <user xsi:type="xsd:string">carlo</user>
            <pwd xsi:type="xsd:string">my_pass</pwd>
            <key xsi:type="xsd:string">server_key</key>
            <dh href="cid:95E080B06256EA1E987E33B6DA0DF698" xsi:type="xsd:Binary"/>
        </upload>
    </soapenv:Body>
</soapenv:Envelope>

= Out message:
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <uploadResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <uploadReturn xsi:type="xsd:string">GriF-carlo-20101004083459</uploadReturn>
        </uploadResponse>
    </soapenv:Body>
</soapenv:Envelope>
=======================================================
```

Figure 6: A typical SOAP message in GriF.

4. A Workflow-enabled Prototype

In order to both illustrate the already mentioned ability of GriF to estimate the type of computing platform best suited for running a complex application built-in into its Ranking feature and to analyze the details of its present implementation, we consider here the calculation of the detailed reactive probabilities of a generic atom-diatom process (see also ref. [46]). In this respect, the modeling via GEMS of atom-diatom chemical reactions is not only key to the progress in plasmas' investigations (because vicarial of unfeasible experimental measurements [47]) but it is also the driving force of the evolution of GEMS and of its ability to both enhance competitive/cooperative massive computational campaigns grounded on QoS and QoU and urge the selective bridging between HTC and HPC platforms.

4.1 The HTC/HPC farm of INTERACTION

The selective (based on an appropriate use of QoS and QoU evaluation tools) bridging of HTC and HPC is illustrated here through its application to the execution of the first (INTERACTION) block of GEMS to compute the potential energy values of atom-diatom systems. To this end use was made of a task farm skeleton distributing the execution of DALTON for a large number of molecular geometries on the HPC platform (squares in Fig. 7) using a moderately high level of theory.

More in detail, when using Workflows in GriF, the user has to upload first a set of instructions (say a kind of main procedure) meant to farm DALTON and upload the related different input files. Then he/she has to combine the Parametric Job modality with the Workflow feature of GriF to run in parallel several subjobs. In the same way, results generated by different subjobs can be channeled via the mentioned main procedure and eventually gathered in the final result file. This means that a general Workflow supported by GriF can make use of all the available programming language constructs and then produce, as already mentioned, complex sequential, conditional and iterative running paths.

In this respect, the present version of GriF acts as a general purpose Science Gateway [48] providing the users with the possibility of running the proper kinds of distribution models [49]. This applies also to GEMS with respect to the criticalities associated with the utilization of the computing resources made by the various programs of the simulator when varying some parameters of the calculations or the dimensionality of the theoretical model used. Typically, at low level of theory, the already mentioned ab initio calculations of the INTERACTION block need not to use the HPC platform.

On the contrary, for higher levels of theory, a queue ensuring at least 1-day long executions per job and having the maximum allowed number of cpus (in order to minimize failure probabilities) was adopted through the already mentioned Ranking option. Then, if the queue is able to run parallel applications by exporting this information [50], it can also be easily used by GriF for running local parallel programs (for example making use of MPI and OpenMP libraries) due to the fact that all the parallel communications between the subjobs will be of the intra-cluster type. In the same way, one has also the possibility of directing the computational tasks to external HPC machines as in the case of the execution of the MPI Grid-oriented version of GAMESS-US package from the Grid to the IBM SP6 nodes of CINECA.

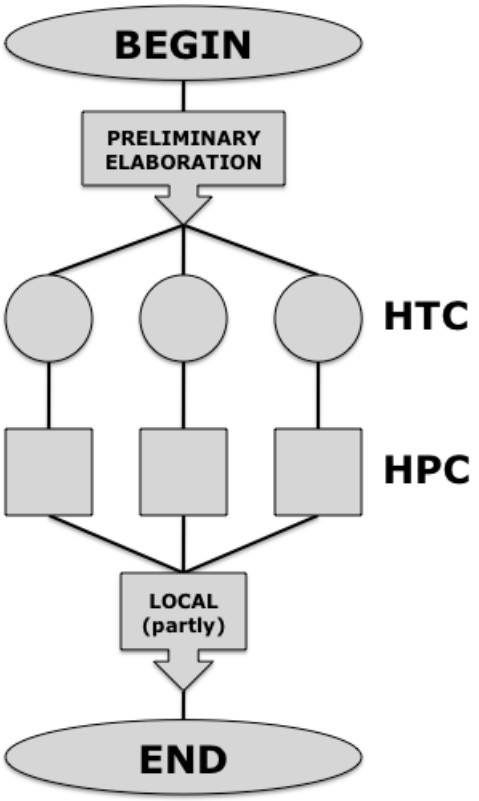

Figure 7: The combined GriF HTC - HPC Workflow (n-n) of GEMS with LOCAL being client activities, HPC being High Performance and HTC being High Throughput Grid jobs.

From the system perspective, at present, each submission is driven by the gLite Workload Management System (WMS) itself which is requested by GriF to execute a devoted pilot Grid job enabled to create, at run-time, a proper SSH bridge to the HPC platform chosen by the user at the beginning. However, on the other hand, the whole procedure is also structured so as to incorporate the advances of the tools offered by the new EMI-2 release [51]. In this way, different submissions on alternative HTC or HPC platforms will be in a completely transparent fashion for the users switched between the two platforms.
In the second block (FITTING) an analytical PES was fitted to the calculated atom-diatom potential energy values. As already mentioned, the fitting needs not to be executed on supercomputers not only because of the algebraical nature of this step of the calculations but also because of its interactive nature (interpolations and graphical representations are continuously needed in the various phases of the procedure).

4.2 The HPC/HTC farm of DYNAMICS

As already mentioned in section 2, the third block of GEMS (DYNAMICS) is the most articulated one and the richest in variety as far as in-house made programs is concerned. In this paper

we pay particular attention to the description of the formalism of the adopted ABC code because of the importance of some of the related parameters for switching from HTC to HPC. Shortly, the computational kernel of the TI ABC program performs the integration of a set of coupled differential equations which in Jacobi coordinates read as follows (for the sake of simplicity we drop here the I label of the considered electronically adiabatic state):

$$\hat{H}_N \xi^{JpK} = \left[\hat{T}_{R,r,\Theta,J,K} + V(R,r,\Theta)\right]\ \xi^{JpK} + C^J_{K,K\pm1}\ \xi^{JpK\pm1} = E\ \xi^{JpK} \tag{4.1}$$

where:

$$\hat{T}_{R,r,\Theta,J,K} = \hat{T}_R + \hat{T}_r + \hat{T}_\Theta + \frac{J(J+1)-2K^2}{2\mu_R R^2} \tag{4.2}$$

$$\hat{T}_R = -\frac{1}{2\mu_R}\frac{\partial^2}{\partial R^2} \tag{4.3}$$

$$\hat{T}_r = -\frac{1}{2\mu_r}\frac{\partial^2}{\partial r^2} \tag{4.4}$$

$$\hat{T}_\Theta = -\left(\frac{1}{2\mu_R R^2} + \frac{1}{2\mu_r r^2}\right)\left(\frac{1}{\sin\Theta}\frac{\partial}{\partial\Theta}\sin\Theta\frac{\partial}{\partial\Theta} - \frac{K^2}{\sin^2\Theta}\right) \tag{4.5}$$

$$C^J_{K,K\pm1} = \frac{\left[J(J+1)-K(K\pm1)\right]^{1/2}\left[j(j+1)-K(K\pm1)\right]^{1/2}}{R^2} \tag{4.6}$$

for all the accessible j, J, K and p values (with j being the diatom rotational quantum number, J the total angular momentum quantum number, K its projection on the space fixed quantization axis, p the parity and μ_r the reduced mass of the reactants). Equations 4.1-4.6 (derived by expanding the nuclear wavefunction Ξ_{NI} in Eq. 2.3 in terms of stationary partial (fixed J) wave functions $\xi^J(R,r,\Theta)$ (in which R, r and Θ are the Jacobi coordinates)) in ABC are converted into the Hyperspherical ones (in which the hyperradius $\rho = \sqrt{R^2+r^2} = \sqrt{R'^2+r'^2}$ acts as continuity variable and the expansion basis for ξ^J is made of the Delves functions $B^{JM}_{vjK}(\alpha,\theta)$ (α and θ are the Delves hyperangles while v and j are the diatom vibrational and rotational quantum numbers (primed for products, unprimed for reactants)) with $\mathbf{g}(\rho)$ being the expansion coefficients matrix). The resulting equations:

$$\frac{\mathrm{d}^2\mathbf{g}}{\mathrm{d}\rho^2} = \mathbf{O}^{-1}\mathbf{U}\mathbf{g} \tag{4.7}$$

in which:

$$O^{v'j'K'}_{vjK} = \langle B^{JM}_{vjK} | B^{JM}_{v'j'K'} \rangle \tag{4.8}$$

and $\mathbf{U}$ the potential and kinetic energy coupling matrix:

$$U^{v'j'K'}_{vjK} = \langle B^{JM}_{vjK} | \frac{2\mu}{\hbar^2}(\bar{H}-E) - \frac{1}{4\rho^2} | B^{JM}_{v'j'K'} \rangle \tag{4.9}$$

(with μ being the reduced mass of the triatom and $\bar{H}$ the ensemble of all the terms of the full Hamiltonian which do not contain derivatives with respect to ρ) are propagated from the initial value ($\rho = 0$) to the asymptotes (large ρ values). When J and/or the number of sectors (N_S) and/or the size of the B basis set (N_B) are large, the ABC computational scheme fits the skeleton shown in Fig. 8.

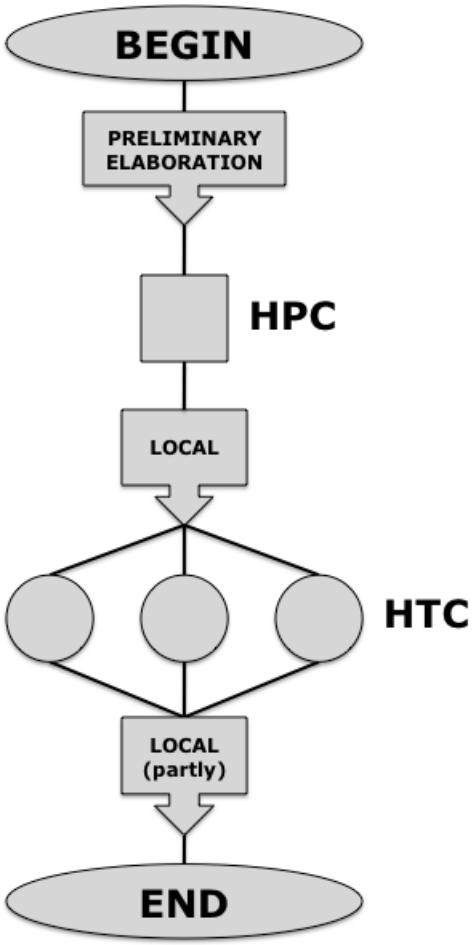

Figure 8: The combined GriF HPC - HTC Workflow (1-n) of GEMS with LOCAL being client activities, HPC being High Performance and HTC being High Throughput Grid jobs.

As a matter of fact, the first section of the code (a HPC job especially for heavy atom-diatom systems) is devoted to the calculation of very large **O** and **U** matrices (requiring the manipulation of several parallel linear algebra routines) in order to assemble the related integration bed. The calculations are not only tightly coupled but they also involve matrices of dimension $N_S N_B^2$ rapidly growing with E and J. For the sake of completeness we comment here that by carrying out an in depth restructuring work the ABC code calculations have been partially decoupled by replicating the evaluation of the Delves functions of adjacent sectors $B^{JM}_{vjK}(\alpha,\theta)$ [52].
The **S** matrix elements (whose square moduli are the state to state reactive probabilities) are then evaluated in the HTC section of the code consisting in the distribution on the Grid of a large number of jobs carrying out each an independent fixed E and J energy propagation of the solution of equations 4.7.

5. Performance and Concurrent Workflows

To validate the procedure, the prototype atom-diatom use case of ref. [46] was considered and the calculations were run for clusters of 100 fixed E and J calculations. GriF chose to distribute the calculations (in which E was varied from 0.4 eV to 1.4 eV using an extremely fine grain of energy values and J was set equal to zero) on 10 Worker Nodes (WN)s of the *gridgate.cs.tcd.ie* Computing Element (CE). For these calculations all internal states vary up to 2.4 eV and rotational quantum numbers up to 50. The reported average execution time for each Grid job run was 947 minutes (with a spread in time between the best and the worst performing WN used in each run of 291 minutes) while the sequential execution time (without job splitting) was 2416 minutes. Therefore, as expected from ref. [7], when using the Parametric option of GriF one can save in time an average of about 60 percent. Then the calculations (exactly the same) were repeated by allowing GriF to adopt the previously illustrated HPC/HTC scheme (as it would be the case when increasing N_S or N_B while keeping unchanged the set of E and J values) on the CINECA machine. The measured average cpu time of the HPC job runs (for a total of 10 runs made of 10 subjobs each) was 233 minutes leading to a saving of about 75 percent in time despite the fact that a job splitting strategy was adopted. On top of the improvement of the performances there is the fact that it was possible to extend the calculation to $J > 0$ without meeting memory size problems (remember that the size of the basis set N_B depends on J). This confirmed the validity of the feature of GriF offering the possibility to resort to HPC machines depending on QoS criteria. Accordingly, this confirmed also the validity of adopting a "Multiple Workflows" approach and properly manage several (up to 500) concurrent Workflows starting from different initial inputs and sharing a common logic. An example enabling, for each Workflow composed by 3 programs called, respectively, `Program-1`, `Program-2` and `Program-3`, conditional and iterative paths, is shown in Fig. 9 under the form of a Shell Script. Moreover, as shown in Fig. 10, the user-friendliness of GriF makes the procedure extremely simple for the users.

```
#!/bin/sh
echo Starting a concurrent Workflow on `hostname` at `date`.
./Program-1
if [ "$?" -gt 20 ]; then
    ./Program-2
    while [ "$?" -gt 0 ]; do
        ./Program-2
    done
else
    ./Program-3
    while [ "$?" -gt 0 ]; do
        ./Program-3
    done
fi
```

Figure 9: Implementing Concurrent Workflows in Grid.

In Fig. 10, in fact, the user first pushes the button ”Upload Your Application” in order to provide the logic common to all Workflows.
In the second step the user, after choosing as Job Type the keyword ’Workflow’, uploads, re-

spectively, a compressed package including the different single or multiple inputs (structured in sub-directories one for each independent Workflow) by using the button ”Select Input”.
In the third step the user can upload a compressed package containing all the binary programs (each of them needs to be statically compiled for the gLite middleware considered here) involved in the computation (shared from each Workflow) by pressing the button ”Upload Programs”. Then when the user pushes the button ”Start” the whole distribution is submitted and the *https* identifier representing the Grid job is returned to the Information area.

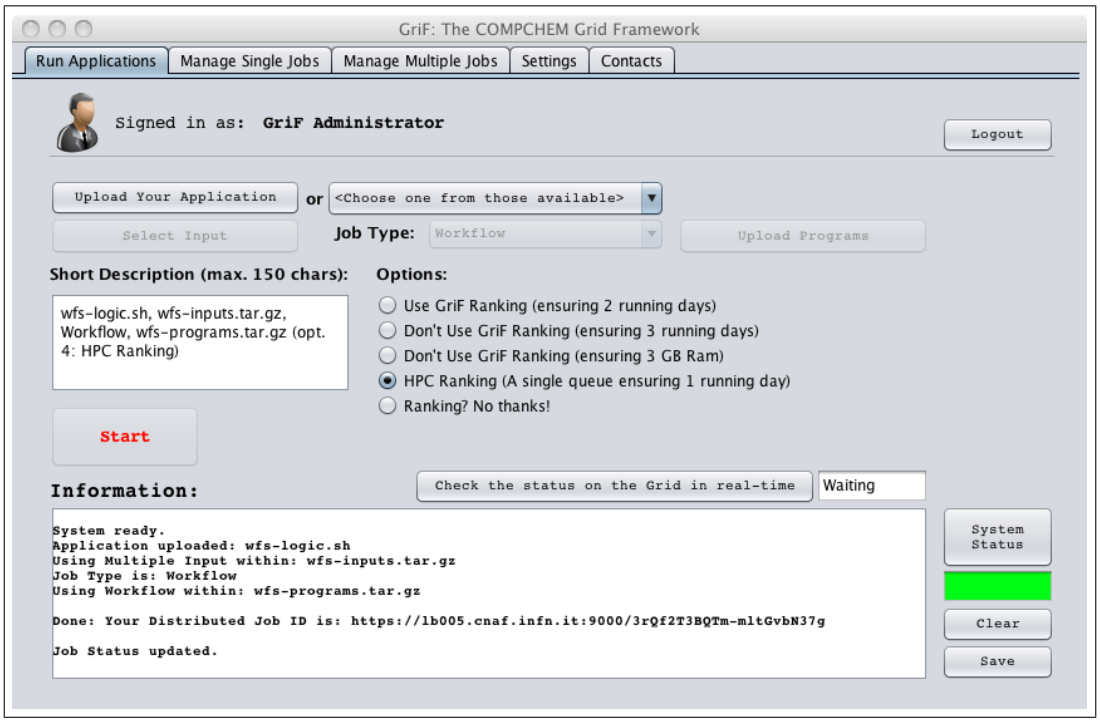

Figure 10: Distributing Parameter Study Workflows on the Grid.

Finally, the user is provided with a view of the management (details associated with a proper use of the ”Manage Multiple Jobs” panel of YC in order to delete, re-schedule and retrieve results for each Workflow (200 in the example given here, running on a different cpu of the same CE queue) are shown in Fig. 11).
In this respect, during the six month period December 2011 - May 2012 COMPCHEM users have run 90 Workflow-based experiments corresponding to a total of 6682 Workflow-based jobs run on the EGI Grid middleware for a total of about 1037 and 101 days of, respectively, wall time and cpu time elapsed and of about 122 GB of consumed real memory.
Moreover, in order to make transparent the various technicalities (like, for example, the necessity of providing the Workflow logic under the form of a Shell Script and to structure as well the various inputs in several directories (typically one for each sub-Workflow)), a new Java Grid Portal mainly devoted to pre-processing the various user needs in order to automatically produce the material required by GriF (e.g. programs and inputs under the form of compressed packages) is being developed.

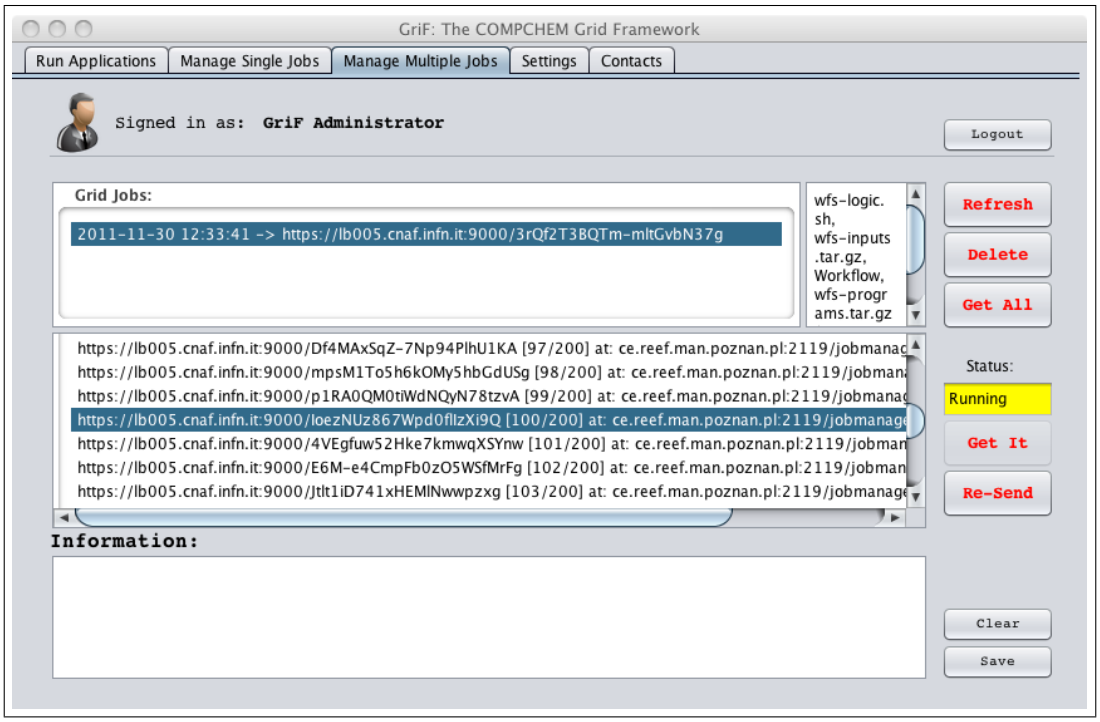

Figure 11: Managing the Parameter Study Workflows distribution.

6. Conclusions

In the present paper the implementation of a version of GriF, a Workflow-oriented Grid Framework structuring computational applications as Web and Grid Services which can be run transparently on different HPC and HTC computing platforms leveraging on the usage of in-house developed QoS and QoU procedures, has been described. To illustrate the mechanisms exploited and the tools developed for this purpose, details of the computational formalism of the dynamics section of a prototype MMST application are given and the relationships between the code parameters and the switch from HTC to HPC platforms are discussed. More specifically, the links between the computational parameters and the use of a HPC/HTC scheme are analyzed to the end of singling out the conditions in which the diverse nature of the workflows would be efficiently managed. Then the structure of the scripts enabling, instead, the composition of serial and parallel tasks in a single complex Workflow and the composition as well of a Workflow of Workflows is illustrated.
All this has been applied to a prototype application of GEMS (the calculation of the efficiency of diatom-diatom reactive processes).
The study points out how the chosen SOA organization of GriF enables the dealing with the distributed (including the inter HTC-HPC communications) implementation of quantum reactive dynamics codes and provides evidence for a user friendly enabling of the joint usage of both HTC and HPC platforms leading to a clear increase in performances even when based on a single Grid Workflow. Results clearly show that GriF, although still in an experimental stage, is well suited for extending the calculations to systems for which the size of memory available on the typical Grid

machines is inadequate and dealing with complex applications by efficiently gathering Workflows of computational tasks on a single composed Workflow by routing them on different parallel and distributed machines and leads to improved performances.
Moreover, a merit of the work reported here is that of pointing out that GriF does not only offer the advantage of producing massive amounts of data to be used as input for the modeling of complex systems and bridging the Grid with supercomputers but it paves the way to the assemblage of higher complexity applications combining different Parameter Studies options and Workflows and to the introduction of a Quality-based economy.

7. Acknowledgements

The research leading to the results presented in this paper has been supported by the use of the Grid resources and services provided by the European Grid Infrastructure (EGI) and the Italian Grid Infrastructure (IGI). For more information, please refer to the EGI-InSPIRE document (http://go.egi.eu/pdnon) and the IGI web server (http://www.italiangrid.it). The authors also acknowledge financial support from the project EGEE III and from the COST CMST Action D37 "CHEMGRID" in particular for building the collaborative distributed network. Thanks are also due for funding to the project "Fundamental Issues on the Aerothermodynamics of Planetary Atmosphere Re-entry" (AO/1-5593/08/NL/HE) and the ESA-ESTEC contract 21790/08/NL/HE, MIUR, ARPA and MASTER-UP srl. Computer time allocation has been obtained through the COMPCHEM VO of EGI.

References

[1] The European Middleware Initiative (EMI); http://www.eu-emi.eu, cited 31 Jul 2012.

[2] The European Grid Initiative (EGI); http://web.eu-egi.eu/documents/other/egi-blueprint, cited 31 Jul 2012.

[3] The EGI-InSPIRE project (Integrated Sustainable Pan-European Infrastructure for Researchers in Europe); http://www.egi.eu/projects/egi-inspire, cited 31 Jul 2012.

[4] The SCI-BUS Project; http://www.sci-bus.eu, cited 31 Jul 2012.

[5] The MAPPER Project; http://www.egi.eu/community/collaborations/MAPPER.html, cited 31 Jul 2012.

[6] GriF: the Grid Framework; http://www.hpc.unipg.it/grif, cited 31 Jul 2012.

[7] Manuali, C., Rampino, S., Laganà, A.: GriF: A Grid Framework for a Web Service Approach to Reactive Scattering, Comp. Phys. Comm., 181, 1179-1185 (2010).

[8] Laganà, A., Riganelli, A., Gervasi, O.: On the Structuring of the Computational Chemistry Virtual Organization COMPCHEM, Lecture Notes in Computer Science, 3980, 665-674 (2006); http://www3.compchem.unipg.it, cited 31 Jul 2012.

[9] Manuali, C., Laganà, A.: A Grid Credit System Empowering Virtual Research Communities Sustainability, Lecture Notes in Computer Science, 6784, 397-411 (2011).

[10] INFN Certification Authority; http://security.fi.infn.it/CA/docs (2010), cited 31 Jul 2012.

[11] Laganà, A.: Supercomputer Algorithms for Reactivity, Dynamics and Kinetics of Small Molecules, Springer Verlag, ISBN 0-7923-0226-5 (1989).

[12] Laganà, A., Riganelli, A.: Reaction and Molecular Dynamics, Springer Verlag, ISBN 3-540-41202-6 (1999).

[13] Italian Grid Infrastructure; http://www.italiangrid.it, cited 31 Jul 2012.

[14] SCAI (Super Computing Applications and Innovation), HPC Department of CINECA; https://hpc.cineca.it/content/about-us, cited 31 Jul 2012.

[15] Gervasi, O., Dittamo, C., Laganà, A.: A Grid Molecular Simulator for E-Science, Lecture Notes in Computer Science, 3470, 16-22 (2005).

[16] Laganà, A., Gervasi, O.: A Priori Molecular Virtual Reality on EGEE Grid, International Journal of Quantum Chemistry, 110, 446-453 (2009).

[17] Manuali, C., Laganà, A., Gervasi, O., Costantini, A.: On the Structuring of a Molecular Simulator as a Grid Service in Chemistry and Material Science Applications on Grid infrastructures, ICTP Lecture Notes Series, ISBN 92-95003-42-X, 24, 63-82 (2009).

[18] Laganà, A.: Towards a grid based universal molecular simulator, in: A. Laganà, G. Lendvay (Eds.): Theory of the dynamics of elementary chemical reactions, Kluwer, Dordrecht, 363-380 (2004).

[19] Gervasi, O., Laganà, A.: SIMBEX: a portal for the a priori simulation of crossed beam experiments, Future Generation Computer Systems, 20, 717-726 (2004).

[20] Nebot-Gil, I., Tel, L. M., Martín, F.: Foundations of Quantum Mechanics in: Andrés, J., Bertrán, J. (Eds.): Theoretical and Computational Chemistry: Foundations, Methods and Techniques, Publications de la Universitat Jaume I, Castelló de la Plana, 1, 1-93 (2007).

[21] Hirst, D. M.: A Computational Approach to Chemistry, Blackwell Science Inc., ISBN 0-632-02431-3 (1990).

[22] The General Atomic and Molecular Electronic Structure System (GAMESS) ab initio quantum chemistry package; http://www.msg.ameslab.gov/gamess, cited 31 Jul 2012.

[23] DALTON, a molecular electronic structure program, Release 2.0 (2005); http://dirac.chem.sdu.dk/daltonprogram.org, cited 31 Jul 2012.

[24] Schatz, G. C.: Fitting Potential Energy Surfaces, in: Computational Reaction and Molecular Dynamics, Laganà, A., Riganelli, A. (Eds.), Lecture Notes in Chemistry, Springer Verlag, 75, 1-30 (2000).

[25] Aguado, A., Tablero, C., Paniagua, M.: Global fit of ab initio potential energy surfaces: I. Triatomic systems, Computer Physics Communications, 108, 259-266 (1998).

[26] Laganà, A., Ochoa de Aspuru, G., Garcia, E.: The largest angle generalization of the rotating bond order potential: three different atom reactions, Journal of Chemical Physics, 108(10), 3886-3896 (1998).

[27] Laganà, A., Riganelli, A.: From Simple Systems and Rigorous Methods to Large Systems and Approximate Methods, in: Computational Reaction and Molecular Dynamics, Laganà, A., Riganelli, A. (Eds.), Lecture Notes in Chemistry, Springer Verlag, 75, 1-12 (1999).

[28] Skouteris, D., Laganà, A., Capecchi, G., Werner, H. J.: Wave packet calculations for the Cl + H_2 reaction, International Journal of Quantum Chemistry, 99, 562 (2004).

[29] Skouteris, D., Castillo, J. F., Manolopoulos, D. E.: ABC: A Quantum Reactive Scattering Program, Comp. Phys. Comm., 133, 128-135 (2000).

[30] VENUS, a general chemical dynamics computer program; http://monte.chem.ttu.edu/group/venus.html, cited 31 Jul 2012.

[31] GROMACS; http://www.gromacs.org, cited 31 Jul 2012.

[32] The DL_POLY Molecular Simulation Package; http://www.cse.scitech.ac.uk/ccg/software/DL_POLY, cited 31 Jul 2012.

[33] Laganà, A., Balucani, N., Crocchianti, S., Casavecchia, P., Garcia, E., Saracibar, A.: An extension of the molecular simulator GEMS to calculate the signal of crossed beam experiments, Lecture Notes in Computer Science, 6784, 453-465 (2011).

[34] Laganà, A., Garcia, E., Paladini, A., Casavecchia, P., Balucani, N.: The last mile of molecular reaction dynamics virtual experiments: the case of the $OH(N=1-10)+CO(j=0-3)$ reaction, Faraday Discussion, DOI: 10.1039/C2FD20046E.

[35] Costantini, A., Murri, R., Maffioletti, S., Laganà, A.: A Grid Execution Model for Computational Chemistry Applications Using the GC3Pie Framework and the AppPot VM Environment, Lecture Notes In Computer Science, 7333, 401-416 (2012).

[36] Skouteris, D., Costantini, A., Laganà, A., Sipos, G., Balaskó, À., Kacsuk, P.: Implementation of the ABC Quantum Mechanical Reactive Scattering Program on the EGEE Grid Platform, Lecture Notes In Computer Science, 5072, 1108-1120 (2008).

[37] Laganà, A., Manuali, C., Costantini, A., Rossi, E., Carpené, M., Ghiselli, A., Cecchi, M.: HIPEG: a project for a high performance grid bridging HTC and HPC in scientific computing, VIRT&L-COMM, 0 (2011).

[38] Manuali, C., Costantini, A., Laganà, A. Cecchi, M., Ghiselli, A., Carpené, M., Rossi, E.: Efficient Workload Distribution bridging HTC and HPC in Scientific Computing, Lecture Notes in Computer Science, 7333, 345-357 (2012).

[39] The Web Services Architecture, W3C Working Group; http://www.w3.org/TR/ws-arch (2004), cited 31 Jul 2012.

[40] Manuali, C., Laganà, A.: GriF: A New Collaborative Framework for a Web Service Approach to Grid Empowered Calculations, Future Generation Computer Systems, 27, 315-318 (2011).

[41] Job Collection - EGEE SEE ROC Wiki; http://wiki.egee-see.org/index.php/Job_Collection, cited 31 Jul 2012.

[42] gLite - Lightweight Middleware for Grid Computing; http://glite.cern.ch, cited 31 Jul 2012.

[43] SOAP Messages with Attachments; http://www.w3.org/TR/SOAP-attachments, cited 31 Jul 2012.

[44] RFC 2111: Content-ID and Message-ID Uniform Resource Locators, http://www.ietf.org/rfc/rfc2111.txt, cited 31 Jul 2012.

[45] Manuali, C.: A Grid Knowledge Management System aimed at Virtual Research Communities Sustainability based on Quality Evaluation, Ph.D. Thesis, Department of Mathematics and Informatics, University of Perugia (IT), 14 Feb 2011; http://www.unipg.it/carlo/PhD_Thesis.pdf, cited 31 Jul 2012.

[46] Rampino, S., Monari, A., Evangelisti, S., Rossi, E., Laganà, A.: A priori modeling of chemical reactions on computational grid platforms: workflows and data models, Chemical Physics, 398, 192-198 (2012).

[47] Capitelli, M., Colonna, G., D'Angola, A.: Fundamental Aspects of Plasma Chemical Physics, Springer Series on Atomic, Optical and Plasma Physics, 66, Springer Science+Business Media, ISBN 978-1-4419-8182-0 (2012).

[48] Specialised Support Centres; http://knowledge.eu-egi.eu/knowledge/index.php/Specialised_Support_Centres, cited 31 Jul 2012.

[49] Foster, I., Kesselman, C.: Scaling System-Level Science: Scientific Exploration and IT implications, Computer, 39(11), 31-39 (2006).

[50] Grid: The GLUE Schema; http://www-numi.fnal.gov/offline_software/srt_public_context/GridTools/docs/glue_schema.html, cited 31 Jul 2012.

[51] EMI-Compute Technical Area; http://www.eu-emi.eu/compute, cited 31 Jul 2012.

[52] Laganà, A., Crocchianti, S., Bolloni, A., Piermarini, V., Baraglia, R., Ferrini, R., Laforenza, D.: Computational granularity and Parallel models to scale up reactive scattering calculations, Computer Physics Communications, 128/1-2, 295-315 (2000).