Virt&l-Comm.2.2012.4

UNIVERSITÀ DEGLI STUDI DI PERUGIA

Dottorato di Ricerca in Matematica ed Informatica per il trattamento dell'informazione e della conoscenza

XXV ciclo

Settore scientifico disciplinare INF/01

SIMULATION MODELING AND WORKFLOW DESIGN

AN APPROACH TO GRID ENABLED DISTRIBUTED APPLICATIONS

Alessandro Costantini

Coordinatore:

Prof. Giulianella Coletti

Relatori:

Prof. Antonio Laganà

Prof. Osvaldo Gervasi

Prof. Sergio Tasso

A.A. 2011/2012

ISSN: 2279-8773

On the cover, the qr-code of COMPCHEM Virtual Organization.

ISSN: 2279-8773

Just a note

Qui di nuovo... non completamente soddisfatto dalle fatiche e dai contenuti del primo Dottorato (lo ricordo, il primo Dottorato è in Scienze Chimiche, Chimica Teorica e Computazionale per la precisione!) e anche un po' perché preso dalla nostalgia dei bei momenti trascorsi, ho deciso di imbarcarmi in questa nuova, quanto conosciuta, avventura. Nuove problematiche, nuovi colleghi, tutto nuovo... o quasi! Si, perché in tutto questo una sola cosa è rimasta inalterata, il gusto e l'emozione che ogni volta provo nel cimentarmi in nuove sfide. Ringrazio per questo i miei supervisori per la possibilità datami, come ringrazio tutti coloro che in questi anni mi hanno supportato (e anche sopportato) nelle scelte come nelle iniziative, con i quali ho discusso (a volte anche aspramente) per la difesa delle rispettive idee e con i quali ho condiviso momenti professionali, ludici e personali. Inutile fare la lista dei nomi, le persone che leggono e che mi conoscono si riconosceranno senza troppi sforzi. Tutto questo ha fatto di me una persona migliore? Non lo so, quello che so è che, di sicuro, oggi sono una persona più dinamica che sa un pochino meglio di ieri cosa vuole fare "da grande". E per concludere, un pensiero a coloro i quali si stanno interrogando sul mio terzo Dottorato. Il destino premia sempre coloro che attendono con fiducia... ma anche no!

Here again... not completely satisfied with the toil and the contents of the first PhD (I remember you, my first PhD is in Chemistry, Computational and Theoretical Chemistry to be precise!) and also taken by the nostalgia of good times gone, I decided to tackle this new, as well as known, adventure. New problems, new colleagues, everything new... or almost! Yes, because after all only one thing is unchanged, the taste and the feeling I experience on new challenges. I have to thank for this my supervisors for the chance given to me, as I thank those people which in recent years have supported me (and also tolerate me) in the choices as in the initiatives, with whom I have discussed (sometimes harshly) for the defense of their ideas and with whom I shared professional, fun and personal time. Needless to make a list of names, people who read and who know myself will recognize themselves without much effort. All of this has made me a better person? I do not know! What I can say is that, of course, today I am a bit more dynamic person that knows a bit better than yesterday what he wants to do tomorrow. And finally, a thought for those who are wondering about my third PhD. The fate always reward those who wait with confidence... or may be not!

ISSN: 2279-8773

Virt&l-Comm.2.2012.4

ISSN: 2279-8773

Contents

ISSN: 2279-8773

ISSN: 2279-8773

ISSN: 2279-8773

Virt&l-Comm.2.2012.4

ISSN: 2279-8773

Abstract

Computing simulations aimed at designing and interpreting the behavior of complex systems are ideally managed through workflows. Workflows [1, 2] are, in fact, tools specifically designed for dealing with data intensive complex applications as those related to scientific research [3]. Scientific simulations, in fact, prompt huge requests of computing resources difficult to match even when using the supercomputers available at large scale computing facilities. At present, a promising alternative to (not always accessible) supercomputers is distributed computing on Grid platforms. The possibility of exploiting Grid technologies and in particular the resources made available by the EGI Grid production infrastructure [4] has allowed us to design a set of workflows associated with the Grid Empowered Molecular Simulator (GEMS) [5] within the activities of the Virtual Organization (VO) COMPCHEM [6].

Accordingly, the thesis is articulated as follows.

The first section deals with the theoretical aspects of computer simulation and the related most used formalisms. In particular, a comparison of the main features of continuous and discrete time representations is presented. The first section continues by discussing probability and statistics methods useful to understand how to model a probabilistic system, how to validate such model and how to design the simulation.

The second section deals with the discussion of the evolution of the ICT technologies from sequential to various forms of concurrency. Then among concurrent technologies the main features of modern distributed computing platforms are discussed. In the same section the articulation of the european production Grid of EGI are illustrated.

The third section deals with the description of the Virtual Organizations assembled within EGI and of the computational workflows developed for some GEMS applications.

In particular, this is the case of the workflows developed for the GFIT3C [7] and ABC [8]applications. GFIT3C is a routine performing the global fitting of the potential energy surface of few atoms. ABC is an atom-diatom quantum mechanical reactive scattering program, for which an automatic

ISSN: 2279-8773

execution procedure has been developed making use of the CG3Pie [9] high throughput framework.

Some other complex applications are also considered in the present thesis such as GROMACS [10] and DL_POLY [11]. For these applications a workflow able to explore different computing platforms has been assembled and results coming from a real multiscale study (involving methane hydrate formation [12]) are presented.

ISSN: 2279–8773

Chapter 1

Computer simulations

1.1 Introduction

The identification and pursuit of Grand Challenges has been a hallmark of the high performance computing arena for well over a decade. In recent years, different technical communities have begun defining Grand Challenge problems for their disciplines. In this way a useful focal point for research and development activities has been provided (in particular for computational modeling and simulation).

Application and methodological areas of modeling and simulation are numerous and diverse:

- Designing and operating transportation systems
- Determining requirements for large scale computer networks
- Reengineering of business precesses
- Designing and analyzing manufacturing systems
- Evaluating military applications and logistic requirements
- Evaluating activities in biological systems
- Designing of new material systems

Yet, in dealing with complex systems, like cellular or cognitive ones, modeling and simulations have often played a role of support to the development of theories and the rationalization of prototype systems rather than predicting the behavior of realistic systems. Only recently efforts of the application area have made a leap forward to effectively simulate real like situations (generating so far applications which can be named virtual situations).

ISSN: 2279-8773

1.2 The foundations of computer simulation

Computer simulation has been often considered an extreme technique to solve hard problems in many different field of research (see for example "Queue Theory" (QT) [13] and "Operation Research" (OR) [14]). Here we give an overview of the most used formalisms in order to provide a picture of the main features of the different approaches and sketch out the role played by simulation in modeling. More specifically we present a technical comparison of the simulation formalisms, and in particular, the difference between continuous and discrete time representations. A simulation experiment consists in the execution of a computer program (the simulation model) with a given set of initial parameters. The outcome of the experiment is a partially ordered sequence of the states of the system, where the concept of state is defined as the set of variables describing the system which are considered relevant for the observer. From the scientist's "gedanken" experiment to the formulation of a computable simulation, the definition of some additional elements is necessary. These elements are the choice of the right time representation, the number and the type of the state variables. This is an important task because it is impossible to work out an exact mapping of the modeller's idea onto a computer program. This gap is better known as the difference between the conceptual model and the simulation model. A simulation model, independently from the formalism in which it is expressed, is always composed by

- a set of parameter variables;
- a set of entities or objects (which can be either structurally present or a volatile, if they are created, live for a time, and are then dropped);
- a set of relations among entities (which, in some cases, are fixed, and in other cases, vary over time);
- a well-defined time representation or formalism.

1.2.1 Systems and Models

Any real-life environment studied by simulation techniques (or for that matter by any other OR model) is viewed as a system. A system, in general, is a collection of entities which are logically related and which are of interest for a particular application. The following features of a system are of interest:

1. Environment: Each system can be seen as a subsystem of a larger system.

ISSN: 2279–8773

2. Interdependency: No activity takes place in total isolation.

3. Sub-systems: Each system can be broken down into sub-systems.

4. Organization: Virtually all systems consist of highly organized elements or components, which interact in order to carry out the function of the system.

5. Change: The present condition or state of the system usually varies over a sufficiently long period of time.

When building a simulation model of a real-life system, one does not simulate the whole system. He/she rather simulates a set of sub-systems related to the problems at hand. This involves modeling parts of the system at various levels of detail. A simulation model is, instead, used in order to study real-life systems which do not currently exist. In particular, one is interested in quantifying the performance of a system under study for various values of its input parameters. Such quantified measures of performance can be very useful for managerial decision processes. All the variables of a system under study can be partitioned into two groups. Variables which are considered as given and are not to be manipulated (*uncontrollable variable*) and those which are to be manipulated till reading a final value named also solution (*controllable variables*). The distinction between controllable and uncontrollable variables mainly depends upon the scope of the study. Another characterization of the relevant variables is whether they are affected or not during a simulation run. A variable whose value is not affected is called exogenous. A variable having a value determined by other variables during the course of the simulation is called *endogenous*. Some of the variables of the system that are of paramount importance are those used to define the status of the system. Such variables are known as *status variables*. These variables form the backbone of any simulation model. At any instance, during a simulation run, one should be able to determine how things stand in the system using these variables. Obviously, the selection of these variables is affected by the kind of information related to system one wants to maintain.

1.2.2 Time representation

Basically, a simulation model is aimed at reproducing real systems, with a different scale of space and time (t). Concerning time, systems can be divided into continuous and discrete ones even when they refer to a continuous time world. As an example, let us consider the dynamics of a post office. The observer may be interested in analysing the behaviour and the evolution of

ISSN: 2279-8773

the queue over time. From this point of view, the system can be considered discrete. In fact, the evolution of the queue is interesting only in some discrete instants of time (for example when a client enters or exits the queue). During the service time, the queue does not change and the simulation model can ignore it. A discrete view of such system is shown in Figure 1.1 where entities (customers) are represented by circles and servers are represented by squares. In most cases systems are intrinsically continuous and they have to be simulated at each time instant. This kind of models are often expressed as sets of Ordinary Differential Equations (ODEs), whose interesting property is to correlate state variables to time. Theoretically, through ODEs it is in principle possible to analytically compute the exact state of the system at each time instant. Most frequently, however, ODEs cannot be analytically solved, and the integration has to be performed numerically. The literature on numerical integration of ODEs offers a wide variety of techniques. All the integration methods have in common the fact that they discretize a continuity variable (most often times that is an ideal candidate for discretization). As a result, the simulation model turns always in a discrete time representation. Here "discrete time" means that the systems change in a synchronous way only at some given instants, just like the model in Figure 1.1. It is important to remember that a discrete representation is always affected by the loss of information. If the system is directly represented with a discrete formalism, the choice of the events considered relevant inevitably excludes other events considered less relevant. When the system is represented with a continuous formalism, the loss of information is due to the discretization introduced by the numerical integration methods. Trying to summarize, a simulation can be referred to two different categories of time representation: the "time step" models and the "discrete event" models. Either approaches are used by the simulator to make a numerical integration of the system. In the "time step" approach the state of the model is updated at regular intervals of time. At any interval, any entities in the system evolve. In this case the lost of precision increases with the size of the time step interval. On the contrary, the discrete event approach alter the system state only in correspondence of some particular events. In this case the lost of precision depends on the level of details introduced by the simulation designer.

Discrete event formalism

Discrete event simulations concern the modeling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time (in mathematical terms we can say that the system is changing a countable numbers of points in time). This points are the ones

ISSN: 2279-8773

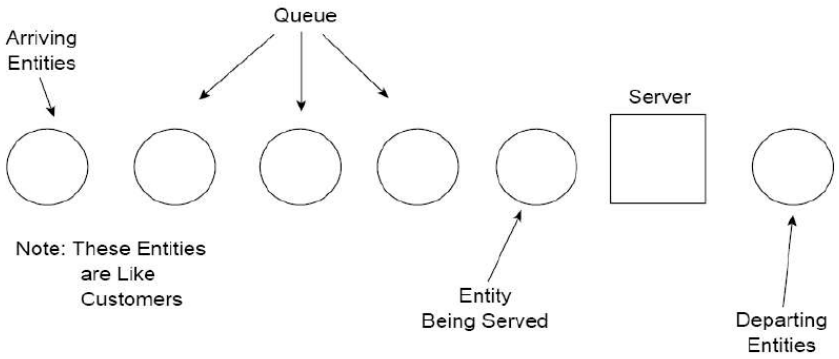

Figure 1.1: A discrete representation of a queue system.

at which an event occurs. Because of the dynamical nature of discrete event simulation models, we must keep track of the current values of the simulated time from one value to another and for such reason we are making use of a special variable (called simulation clock) which gives the current value of the time in the simulated system. When the model is written in a general purpose language (like FORTRAN or C), the time units are assumed to be the same on the input parameters. Due to historical reasons, two principal approaches have been suggested for advancing the simulation clock:

- next event time advanced: the simulation clock is initialized to zero and the times of occurrence of future events are determined. The simulation clock is then advanced to the time of occurrence of the imminent future event and the system state is updated to account for the event that has occurred. Then the simulation clock is advanced to the time of the new imminent event, the state of the system is updated and the future event times are determined. This process of advancing the simulation clock from one event time to another is repeated until the predetermined simulation stopping condition has been reached.

- fixed increment time advance: in this case the time is advanced in fixed intervals (every t time units)

The fixed increment time advance approach has 2 disadvantages:

1. it can only detect events that occur during interval $(t_0, t_0 + t]$ at time (t_0+t), thereby introducing errors in the simulation unless events occur exectly at the beginning or end of the interval;

ISSN: 2279-8773

2. if the interval between two events is very large compared to t, then the simulation goes through several unproductive clock increments.

The next event time advanced approach has the disadvantage of requiring more information to be stored. Moreover the simulation time does not flow smoothly. Even with those limitations, the next event time advanced approach is used in the majority of discrete event simulation languages.

The time continuous formalisms

The continuous time representation is based on a set of differential equations describing the behaviour of some variables over time. Therefore a continuous simulation model can be reconducted to a numerical integration of the related set of equations. In order to adopt the appropriate computation method, it is necessary to group analytical models into two different categories: the causal and non-causal models. A typical analytical model is expressed by a system of equations, which can be often solved analytically and sometimes only numerically. Since the simulation of such systems is strictly based on numerical integration methods, its solution algorithms depend on the type of causal relations established among variables. An equation system like the one in Figure 1.2. can be hierarchically decomposed, in order to determine the sequence of dependencies.

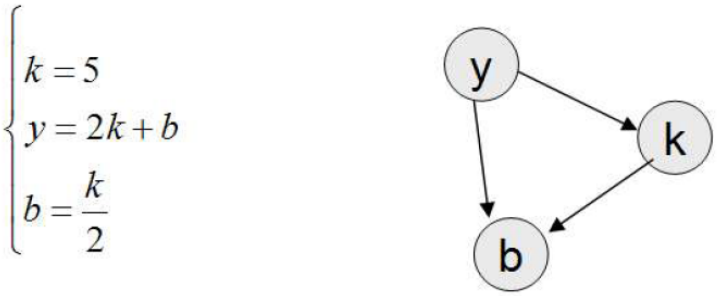

Figure 1.2: The sequential representation of an equation system.

The implicit causal relations can be explicitly described by the following relationships:

- y depends on k, b
- b depends on k

- k is independent.

In order to simulate this system the simulation engine is able to sort the equations according the causal relation diagram and so computes the variables in the right order. On the contrary, when a model cannot be sorted, due to a dependency cycle or algebraic loops, it is considered a non-causal model. The numerical computation of non-causal set of equations is complicated, since they must be transformed into an equivalent set of causal relations. Once the continuous model is expressed as a causal and ordered set of equations it can be numerically solved by a computer. The most used algorithms to do the numerical integration are the Euler [15] and the Runge-Kutta [16] ones. The integration method introduced by Leonard Euler is the simplest one. It is based on the assumption that the functions remain constant during the time interval dt. This method is simple and adequate to simulate the most part of the continuous simulation models, even if it may be affected by a significant error when the time interval is not short enough. In this case other more precise methods can be used.

Among these the Runge-Kutta algorithm is the most popular and used. It finds a better approximation of the rate between t and $t + dt$, computing the average of the rates at t and at $t + dt$, which has been previously estimated with the Euler method. Although the Runge-Kutta method offer the advantage of reducing the integration error it may also alter the effect of sudden shocks, deliberately introduced by the modeller (random noise, step or pulse functions, and so forth).

1.2.3 The Discrete EVent System

The Discrete Event System Specification (DEVS) [17] method integrates the continuous and discrete time formalisms in a unified discrete event based approach. DEVS provides a means for specifying a mathematical object called a system. Basically, a system has a time base, inputs, states, and outputs, and functions for determining next states and outputs given the current states and inputs. Discrete event systems represent certain constellations of such parameters just as continuous systems do. For example, the inputs in discrete event systems occur at arbitrarily spaced moments, while those in continuous systems are piecewise continuous functions of time. The insight provided by the DEVS formalism consists of the simple way it characterizes how discrete event simulation languages specify discrete event system parameters. On the ground of this abstraction, it is possible to design new simulation languages with sound semantics that is easier to understand. The conceptual framework underlying the DEVS formalism is shown in Figure

ISSN: 2279-8773

1.3 where the model for the ping-pong game is represented. Here each player (A and B) of Fig. 1.3 bears two events: the input event (receive), and the output event (send); and two states: Send and Wait. Send state takes 0.1 seconds to send back the ball (that represents the output event), while Wait maintains the state until the player receives the ball (that represents the input event). The two players are connected by the relationship: the output event of A is transmitted as input event to player B, and vice versa.

The modeling and simulation activity concerns three basic objects:

Model, which is a set of instructions for generating data comparable to that observable in the real system. The structure of the model is its set of instructions. The behavior of the model is the set of all possible data that can be generated by faithfully executing the model instructions.

Simulator, which exercises the model's instructions to actually generate its behavior.

Experimental frame, which captures how the modeler's objectives impact on model construction, experimentation and validation.

These basic objects are related by two relationships:

Modeling relationship, linking real system and model, defines how well the model represents the system or entity being modeled. In general terms a model can be considered valid if the data generated by the model agrees with the data produced by the real system in an experimental frame of interest.

Simulation relationship, linking model and simulator, represents how faithfully the simulator is able to carry out the instructions of the model.

The basic items of data produced by a system or model are time segments. These time segments are mappings from intervals defined over a specified time base to values in the ranges of one or more variables. The variables can either be observed or measured and the structure of a model may be expressed in a mathematical language called formalism focusing on the changes of variable values and generates time segments that are piecewise constant. Thus an event is a change in a variable value, which occurs instantaneously.

The DEVS formalism

A Discrete Event System Specification is a structure

ISSN: 2279-8773

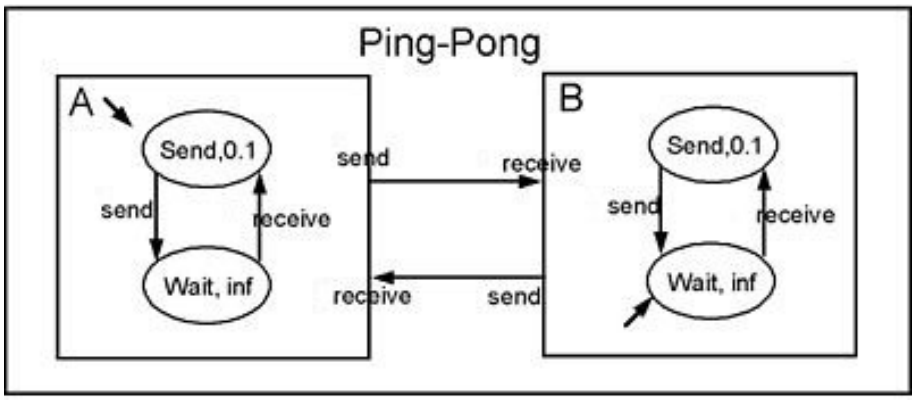

Figure 1.3: Representation of DEVS Model for Ping-Pong Game. Each player, A and B, has his events and his states. The two players are connected by relations.

$$M =< X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta > \tag{1.1}$$

where X is the set of inputs, S is a set of states, Y is the set of outputs $\delta_{int} : S \to S$ is the internal transition function, $\lambda : S \to Y$ is the output function, $ta : S \to R^{+}_{0,\infty}$ is the time advance function and $\delta_{ext} : Q \times X \to S$ is the external transition function where

$$Q = \{(s, e)|s \in S, 0 \leq e \leq ta(s)\} \tag{1.2}$$

is the total state set and e is the time elapsed since last transition. The interpretation of these elements is illustrated in Figure 1.4. At any time the system is in some state, S. If no external event occurs the system will stay in state S for time $ta(s)$. Notice that $ta(s)$ could be a real number as one would expect. But it can also take the values 0 and ∞. In the first case, the stay in state S is so short that no external events can intervene (we say that S is a transitory state). In the second case, the system will stay in S forever unless an external event interrupts its slumber. We say that S is a passive state in this case. When the resting time expires, i.e., when the elapsed time, e, is equal to $ta(s)$, the system outputs the value, $\lambda(s)$, and changes to state $\lambda_{int}(s)$. Output is only possible just before internal transitions. If an external event $x \in X$ occurs before this expiration time, i.e., when the system is in total state (s, e) with $e \leq ta(s)$, its state changes to $\delta_{ext}(s, e, x)$. Thus the internal transition function dictates the system's new state when no events have occurred since the last transition. While the external transition function dictates the system's new state when an external event occurs, this

ISSN: 2279-8773

state is determined by the input, x, the current state, S, and how long the system has been in this state, e, when the external event occurred. In both cases, the system is then is some new state S' with some new resting time, $ta(s')$.

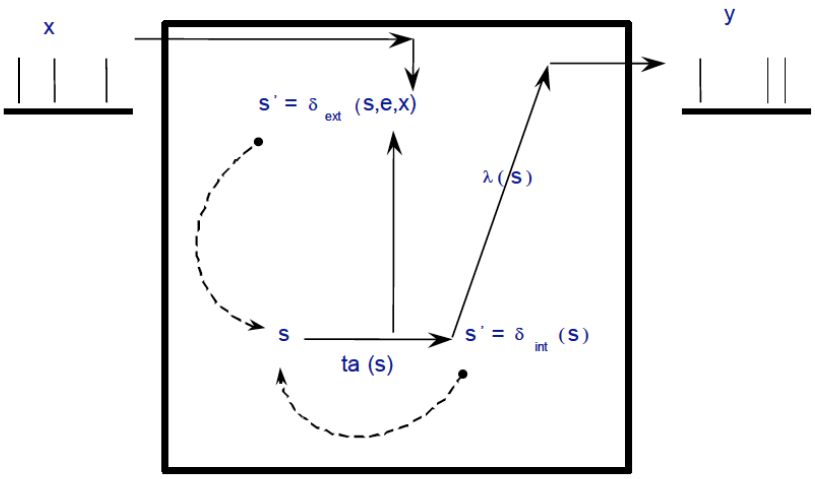

Figure 1.4: Representation of the DEVS elements as decribed by eqs. 1.1 and 1.2

Basic models and connections

To be more specific, modular discrete event models require that we adopt a view different from that fostered by traditional simulation languages. In fact, in modular specification in general, a model has to be viewed as possessing input and output ports through which the interaction with the environment is mediated. In the discrete event case, events determine values appearing on such ports. More specifically, when external events, arising outside the model, are received on its input ports, the model description must determine how to respond to them. Internal events too, arising within the model change its state, as manifest themselves as events on the output ports to be transmitted to other model components.

A basic model contains the following information:

- the set of input ports through which external events are received,
- the set of output ports through which external events are sent,

- the set of state variables and parameters: two state variables are usually present, *phase* and *sigma* (in the absence of external events the system stays in the whole current *phase* for the time given by *sigma*),

- the time advance function which controls the timing of internal transitions (when the *sigma* state variable is present, this function just returns the value of *sigma*),

- the internal transition function which specifies to which next state the system will transit after the time given by the time advance function has elapsed,

- the external transition function which specifies how the system changes state when an input is received. In this case the effect is to place the system in a new *phase* and *sigma* thus scheduling it for a next internal transition; the next state is computed on the basis of the present state, the input port and value of the external event, and the time that has elapsed in the current state,

- the confluent transition function which is applied when an input is received at the same time that an internal transition is to occur,

- the output function which generates an external output just before an internal transition takes place.

Basic models may be coupled in the DEVS formalism to form a coupled model. A coupled model tells how to couple (connect) several component models together to form a new model. This latter model can itself be employed as a component in a larger coupled model, thus giving rise to hierarchical construction. A coupled model contains the following information:

- the set of components,

- the set of input ports through which external events are received,

- the set of output ports through which external events are sent.

These components can be synthesized together to create hierarchical models having external input and output ports. The coupling specification consisting of:

- the external input coupling which connects the input ports of the coupled to model to one or more of the input ports of the components,

ISSN: 2279-8773

- the external output coupling which connects output ports of components to output ports of the coupled model thus when an output is generated by a component it may be sent to a designated output port of the coupled model and thus be transmitted externally,

- the internal coupling which connects output ports of components to input ports of other components when an input is generated by a component it may be sent to the input ports of designated components (in addition to being sent to an output port of the coupled model)

On the same way, a coupled model can be expressed as an equivalent basic model in the DEVS formalism. Such a basic model can itself be employed in a larger coupled model. This shows that the formalism is closed under coupling as required for hierarchical model construction. Expressing a coupled model as an equivalent basic model captures the means by which the components interact to yield the overall behavior.

1.2.4 Alternative approaches to computing simulation

Since the early days of simulations, people have constantly looked for new and better ways to model a system, as well as a novel way to use existing computer hardware and software in simulation.

Parallel and distributed simulations

Usually, all the sequential simulations operates basically using the same approach: a simulation clock and event list interact to determine which event will be processed next, the clock is advanced to the time of this event and the computer executes the event logic (i.e. updating state variables, manipulating lists for queues and event, generating random numbers, collecting statistics). This logic is executed sequentially and the simulation is done in a single computer. In recent years computer technology has enabled individual computers or processors to be linked together to *parallel* or *distributed* computing environments (well described in chapter 2). In such environment, it may be possible to *distribute* different tasks of a computing part across individual processors (that may belong or not to the same computing node) operating at the same time, or in *parallel*, and thus reduce the overall time to complete the task.

There are normally four principal benefits to executing a simulation program across multiple processors:

ISSN: 2279-8773

Reduced execution time By subdividing a large simulation computation into many subcomputations, and executing the subcomputations concurrently across different processors, one can reduce the execution time. In computer simulations it may be necessary to reduce execution time and avoid to wait long periods of time to receive results produced by the simulation. Alternatively, when used to create a virtual world into which humans will be immersed, multiple processors may be needed to complete the simulation computation fast enough so that the simulated world evolves as rapidly as real life. This is essential to make the computer-generated world "look and feel" to the user just like the real thing.

Geographical distribution Executing the simulation program on a set of geographically distributed computers enables one to create virtual worlds with multiple participants that are physically located at different sites.

Integrating simulators on different machines Suppose that flight simulators for different types of aircraft have been developed by different manufacturers. Rather than porting these programs to a single computer, it may be more cost effective to "hook together" the existing simulators, each executing on a different computer, to create a new virtual environment. Again, this requires the simulation computation to be distributed across multiple computers.

Fault tolerance Another potential benefit of utilizing multiple processors is the increased tolerance to failures. If one processor goes down, it may be possible for another processor to pick up the work of the failed machine, allowing the simulation computation to proceed despite the failure. By contrast, if the simulation is mapped to a single processor, failure of that processor means the entire simulation must stop.

Web-Based simulations

With the rapid development of the Internet and the World Wide Web (WWW), a natural question arises as to how this gigantic network might be used to build, share, modify, distribute and run simulations. During the years, a wide range of issues have arisen in this regard, including client-server arrangements, for increasing processing power, dissemination for simulation models and result, publication, education and training. While it is difficult to predict precisely how the internet and the web might affect simulation, it seems clear that the interest is high and that many people are exploiting a

ISSN: 2279-8773

variety of ways to use this technology in novel ways to support simulations. For example, the combination of WWW and Java [18] offers a set of unique capabilities for computing [19] including transparency of network heterogeneity, transparency of operating system heterogeneity, and transparency of user interface heterogeneity. Web technology has the potential to significantly alter the ways in which simulation models are developed, documented, and analyzed. The web represents a new way for both publishing and delivering multimedia information to the world. Web technologies that could be leveraged in a simulation support role are numerous. For example, the Hypertext Markup Language (HTML) [20] provides both document formatting and direct linkage to other documents. This capability can significantly improve the information acquisition and presentation process for model developers. Moreover, the power of hypermedia available on the web can also influence the way models are developed and used. The forms capability of HTML provides a simple mechanism to construct a graphical user interface that could be used in web-based simulation modeling environments. The WWW can also act as a software delivery mechanism and distributed computation engine. Three approaches for web-based can be identified in the literature [21]. Server-hosted simulation allows existing simulation tools to be hosted on a web server and accessed by clients via normal HTML pages. This approach has the advantage of using a familiar tool and enables the reuse of existing models. The disadvantage is that the communication power provided by animation in these tools is not visible over the web. Client-side simulation allows simulation tools to use an applets-based approach that minimizes the learning curve, but on the expense of power and flexibility. The performance of this type of simulation is also limited by the client machine capabilities. Java is an example of programming languages that may be used to develop simulation applications that may be executed on the client machine. Several Java based simulation packages and languages have been developed. For example, Simjava [22] uses a basic discrete event simulation engine and extends this with Java's graphical user interface features. JSIM is a Java-based simulation and animation environment based on both the process interaction and event scheduling approaches to simulation. It includes a graphical designer that allows for graphical model construction on the web [23]. Silk [24] is a process-based multithreaded simulation language built in Java. Hybrid client/server simulation attempts to combine the advantages of server hosted and client executed simulations. The approach relies on hosting the simulation engine on the server and using Java for visualization of the animation to provide a dynamic view on the client machine.

ISSN: 2279-8773

1.3 Statistics and Probability concepts

The completion of a successful simulation study involves much more than constructing a flowchart of the system under study, translating the flowchart into a computer program and then making replications of each proposed system configurations. As an example, accounting for the variation of observed variables as well as the random sampling of observed variables themselves when it is impossible to represent them exaustively, prompt the use of statistics and probability. As a matter of facts, the use of probability and statistics to chose inputs, to validate the model and to design the simulation experiments is an integral part of a simulation study.

1.3.1 Random numbers

In a simulation, random numbers generation [25] is used to the end of making the simulation more realistic by mimicking some relevant theoretical or empirical distributions. This is usually pursued by generating a finite uniformly distributed set of numbers otherwise known as pseudo-random numbers. Pseudo-random numbers can be used either as such or to generate random numbers from different theoretical or empirical distributions, known in general as random variates or stochastic variates. There are two main methods to generating random numbers. In the first approach, a physical phenomenon is used as a source of randomness from where random numbers can be generated. Random numbers generated in this way are called true random numbers. A true random number generator [25] requires a completely unpredictable and nonreproducible source of randomness. Such sources can be found in nature, or they can be created from hardware and software. For instance, the elapsed time between emissions of particles during radioactive decay is a well known randomized source. Also, the thermal noise from a semiconductor diode or resistor can be used as a randomized source. Finally, sampling human computer interaction processes, such as keyboard or mouse activity of a user, can give rise to a randomized source. True random numbers are ideal for critical applications, such as cryptography, due to their definitely unpredictable and realistic behaviour. However, they are less useful in computer simulation, where as will be seen in the following, where we need to be able to reproduce a given sequence of random numbers. In addition, despite their several attractive properties, the production and storing of true random numbers is very costly. On the contrary the generation of pseudo-random numbers, which is after all the most popular approach, is to use a mathematical algorithm. Efficient easy to implement and use pseudo-random numbers generation algorithms have been developed

ISSN: 2279-8773

in the past. These algorithms produce numbers in a deterministic fashion. That is, given a starting value (known as the seed) the same sequence of random numbers can be produced each time the same seed is used. Despite its deterministic generation, the generated set of numbers is uniformly distributed and does not contain repetition for a predetermined size of numbers as well as other statistical tests. An advantage of generating pseudo-random numbers in a deterministic fashion is that they are reproducible. In fact, the same sequence of random numbers is produced each time the pseudo-random generator is run providing the same seed. This is helpful when debugging a simulation program, as we typically want to reproduce the same sequence of events in order to verify the accuracy of the simulation. Pseudo-random numbers (and, in general, random numbers) are typically generated on demand. That is, each time a random number is required, the appropriate generator is called returning a random number. Consequently, there is no need to generate a large set of random numbers in advance and store them in an array for future use.

1.3.2 Random numbers generators

In general, an acceptable method for generating random numbers must yield sequences of numbers or bits that are uniformly distributed, statistically independent, reproducible, and non-repeating for any desired length. Historically, the first method for creating random numbers by computer was Von Neuman's mid-square method [26]. His idea was to take the square of the previous random number and to extract the middle digits. The mid-square method was relatively slow and statistically unsatisfactory. It was later abandoned in favour of other algorithms as the congruential method, the Tausworthe generators, the lagged Fibonacci generators, and the Mersenne twister.

The congruential method

The advantage of this congruential method [27] is that it is extremely simple, fast, and produces pseudo-random numbers statistically acceptable for computer simulations. The congruential method uses the following recursive relationship to generate random numbers.

$$x_{i+1} = ax_i + c(mod\ m) \tag{1.3}$$

where x_i, a, c and m are all non-negative numbers. Given that the previous random number was x_i, the next random number x_{i+1} can be generated by multiplying x_i by a and then added to c. Then, the modulus m of the result. That is, divide the result by m and set x_{i+1} equal to the remainder

ISSN: 2279–8773

of this division. Of this method, known as the mixed congruential method, a simpler variation (the multiplicative congruential method) utilizing the relationship $x_{i+1} = ax_i(mod\ m)$ can be used. The numbers generated by a congruential method fall in the interval ranging from 0 and $m-1$. Uniformly distributed random numbers between 0 and 1 can be obtained by simply dividing the resulting x_i by m. The number of successively generated pseudo-random numbers after which the sequence starts repeating itself is called the period. If the period is equal to m, then the generator is said to have a full period. Theorems from number theory show that the period depends on m. The larger the value of m, the longer is the period. In order to get a generator started, we further need an initial seed value for x. The mixed congruential method described in eq. 1.3 can be thought of as a special case of a following generator:

$$x_{i+1} = f(x_i, x_{i-1},\ \dots)\ (mod\ m), \tag{1.4}$$

where $f(\dots)$ is a function of previously generated pseudo-random numbers. A special case of the above general congruential method is the quadratic congruential generator. This has the form:

$$x_{i+1} = a_1 x_i^2 + a_2 x_{i-1} + c\ (mod\ m) \tag{1.5}$$

The special case of $a_1 = a_2 = 1$, $c = 0$ and m being a power of 2 has been found to be related to the midsquare method. Another special case that has been considered is the additive congruential method, which is based on the relationship:

$$f(x_i, x_{i-1}, \dots, x_{i-k}) = a_1 x_i + a_2 x_{i-1} + \dots + a_k x_{i-k}. \tag{1.6}$$

Tausworthe generators

Tausworthe generators [28] are additive congruential generators obtained when the modulus m is equal to 2. In particular,

$$x_i = (a_1 x_{i-1} + a_2 x_{i-2} + \dots + a_n x_{i-n})(mod\ 2) \tag{1.7}$$

where x_i can be either 0 or 1. This type of generator produces a stream of bits $\{b_i\}$. In view of this, it is sufficient to assume that the coefficients a_i are also binary. Thus, x_i is obtained from the above expression by adding some of the preceding bits and then carrying out a modulo 2 operation. This is equivalent to an exclusive OR operation, notated as $\oplus$ in Table 1.1.

$A \oplus B$ is true (i.e. equal to 1), when either A is true and B false, or A is false and B true. Tausworthe generators are independent of the computer used and its word size and have very long cycles. However, they are too slow since they only produce bits.

ISSN: 2279–8773

Table 1.1: Truth table

A	B	A $\oplus$ B
0	0	0
0	1	1
1	0	1
1	1	0

The Lagged Fibonacci Generators

The Lagged Fibonacci Generators (LFG) [29] are an important improvement over the congruential generators, and they are widely used in simulation. They are based on the well-known Fibonacci sequence, an additive recurrence relationship, whereby each element is obtained using the two previously computed elements, as shown below

$$x_n = x_{n-1} + x_{n-2} \tag{1.8}$$

where $x_0 = 0$ and $x_1 = 1$. The beginning of the Fibonacci sequence is: 0, 1, 1, 2, 3, 5, 8, 13, 21. Based on this recurrence relation, the general form of LFG can be expressed as follows:

$$x_n = x_{n-j}\ O\ x_{n-k}\ (mod\ m) \tag{1.9}$$

where $0 < j < k$, and appropriate initial conditions have been set. In this generator, the next element is determined by combining two previously calculated elements which lag behind the current element utilizing an algebraic operation O. This operation O can be either an arithmetic operation (addition, subtraction, multiplication) or a binary operation XOR. If O is an addition, then this LFG is called the Additive LFG (ALFG) [29]. Likewise, if O is a multiplication, then the LFG is called the Multiplicative LFG (MLFG) [29]. The additive LFG is the most frequently used generator. In this case, the next element is calculated as follows:

$$x_n = x_{n-j} + x_{n-k}\ (mod\ M) \tag{1.10}$$

where $0 < j < k$. As can be easily seen, it is very simple to implement and quite fast to execute. A very long period, equal to m_{k-1}, can be obtained if m is a prime number. However, using a prime number make the execution slower. Thus, typically m is set to 2^{32} or 2^{64}. In this case the maximum

ISSN: 2279–8773

period of the additive LFG is $(2^k - 1)x2^{m-1}$. The multiplicative LFG is accordingly:

$$x_n = x_{n-j} * x_{n-k} \ (mod\ m) \tag{1.11}$$

where m is set to 2^{32} or 2^{64} and $0 < j < k$. The maximum period is $(2^{k-1}) * 2^{m-3}$. In general, LFGs generate a sequence of random numbers with very good statistical properties, and they are nearly as efficient as the linear congruential generators. Their execution can also be parallelized. However, LFGs are highly sensitive on the seed. That is, the statistical properties of an output sequence of random numbers varies from seed to seed. Determining a good seed, however, for LFGs is a difficult task.

The Mersenne twister

The Mersenne twister (MT) [30] is an important pseudo-random number generator with superior performance. Its maximum period is $2^{19937}-1$, which is much higher than many other pseudo-random number generators, and its output has very good statistical properties. The MT generates a sequence of bits, which is as large as the period of the generator after which it begins to repeat itself. This bit sequence is typically grouped into 32-bit blocks (i.e., blocks equal to the computer word). The blocks are considered to be random. The following is the main recurrence relationship for the generation of random sequence of bits:

$$x_{k+n} = x_{k+m} \oplus (x_k^u | x_{k+1}^l)A, k \geq 0 \tag{1.12}$$

Assuming that each block, represented by x, has a size of w bits, x_k^u is the upper w-r bit of x_k,where $0 \leq r \leq w$, x_{k+1}^l is the lower bit of x_{k+1}, $\oplus$ the exclusive OR, | indicates the concatenation (i.e., joining) of two bit strings, n the degree of recurrence relation, m the integer in the range of $1 \leq m \leq n$, u and l are the additional Mersenne Twister tempering bit shifts and A is a wxw matrix defined as below so that the multiplication operation in the above recurrence can be performed extremely fast

$$\begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ a_{w-1} & a_{w-2} & a_{w-3} & \cdots & a_0 \end{pmatrix}$$

The recurrence relation is initialized by providing seeds for the first d blocks, i.e., x_0, x_1, ..., x_{n-1}. The multiplication operation xA can be made

ISSN: 2279–8773

very fast as follows:

$$xA = \begin{cases} x \gg 1 & \text{if } x_0 = 0 \\ (x \gg 1) \oplus a & \text{if } x_0 = 1 \end{cases}$$

where $a = \{a_{w-1}, a_{w-2}, \ldots, a_0\}$ and $x = \{x_{w-1}, x_{w-2}, \ldots, x_0\}$. At the last state of the algorithm, in order to increase the statistical properties of generator's output, each generated block is multiplied from the right with a special wxw invertible tempering matrix T. This multiplication is performed as that with matrix A (see above) and it involves only bitwise operations, as follows.

$$\begin{array}{rcll} y & = & x & \oplus \quad (x \gg u) \\ y & = & x & \oplus \quad ((x \ll s) \wedge b) \\ y & = & x & \oplus \quad ((x \ll t) \wedge c) \\ y & = & x & \oplus \quad (x \gg l) \end{array} \tag{1.13}$$

where b and c are block size binary bitmasks (vector parameters), l, s, t and u are pre-determined integer constants and the $\ll$, $\gg$ indicates the bitwise left and right shifts.

1.3.3 Stochastic variates

Uniformly distributed pseudo-random numbers are normally referred to as random numbers and when they follow a specific distribution are called random variates or stochastic variates. The most commonly used technique for generating random variates is the inverse transformation method. This method is applicable only to cases in which the cumulative density function can be inverted analytically. Assume that we wish to generate stochastic variates from a Probability Density Function (PDF) $f(x)$ [31]. Let $F(x)$ be its cumulative density function. We note that $F(x)$ is defined in the region [0,1]. We explore this property of the cumulative density function to obtain the following simple stochastic variates generator. We first generate a random number r which we set equal to $F(x)$. The quantity x is then obtained by the inverse transformation of F as follow:

$$x = F^{-1}(r) \tag{1.14}$$

1.3.4 Sampling from some probability distributions

In this section, inverse transformation method has been applied to generate variates from most common probability distributions giving some examples.

ISSN: 2279–8773

Sampling from a uniform distribution

The probability density function of the uniform distribution [32] is defined as follows (see Figure 1.5):

$$f(x) = \begin{cases} \frac{1}{b-a} & a < x < b \\ 0 & \text{otherwise} \end{cases}$$

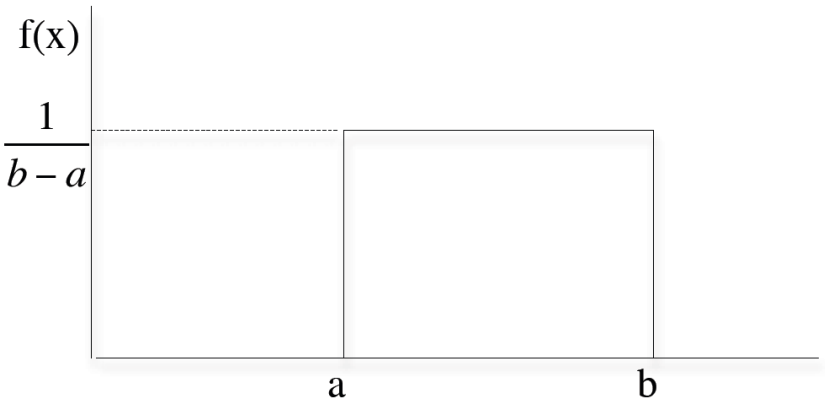

Figure 1.5: Graphical representation of the uniform distribution.

The cumulative density function is:

$$F(x) = \int_a^x \frac{1}{b-a} \mathrm{d}t = \frac{x-a}{b-a} \tag{1.15}$$

The expectation and variance are given by the following expressions:

$$E(X) = \int_a^b f(x) x \mathrm{d}x = \frac{1}{b-a} \int_a^b x \mathrm{d}x = \frac{b+a}{2} \tag{1.16}$$

and

$$Var(X) = \int_a^b (x - E(X))^2 F(x) \mathrm{d}x = \frac{(b-a)^2}{12} \tag{1.17}$$

The inverse transformation method for generating random variates is as follows

$$r = F(x) = \frac{x-a}{b-a} \tag{1.18}$$

with $x = a + (b-a)r$.

ISSN: 2279-8773

Sampling from an exponential distribution

The probability density function of the exponential distribution [32] is defined as follows:

$$f(x) = ae^{-ax}, a > 0, x \geq 0 \tag{1.19}$$

Starting from eq.1.19, the cumulative density function is

$$F(x) = \int_0^x f(t)\mathrm{d}t = \int_0^x ae^{at}t\mathrm{d}t = 1 - e^{-ax} \tag{1.20}$$

The expectation and variance are given by the following expressions

$$E(X) = \int_0^\infty aet^{-at}\mathrm{d}t \ \mathrm{d}t = \frac{1}{a} \tag{1.21}$$

and

$$Var(X) = \int_0^\infty (t - E(X))^2 e^{-at}t\mathrm{d}t \ \mathrm{d}t = \frac{1}{a^2} \tag{1.22}$$

By applying the inverse transformation method for generating random variates one obtains

$$r = F(x) = 1 - e^{-ax} \tag{1.23}$$

or

$$x = -\frac{1}{a}log(1 - r) \tag{1.24}$$

Since $1 - F(x)$ is uniformly distributed in [0,1], the equation becomes $r = e^{-ax}$ and this leads to $x = -\frac{1}{a}log\ r$.

Sampling from a normal distribution

A random variable X with probability density function

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}, \ -\infty < x < +\infty \tag{1.25}$$

where σ is positive, is said to have a normal distribution with parameters μ and σ. The expectation and variance of X are μ and σ^2 respectively. If $\mu = 0$ and $\sigma = 1$, then the normal distribution is known as the standard normal distribution and its probability density function is as follow

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}x^2}, \ -\infty < x < +\infty \tag{1.26}$$

ISSN: 2279-8773

If a random variable X follows a normal distribution with mean μ and variance σ^2, then the random variable Z can be defined as follows

$$Z = \frac{X - \mu}{\sigma}. \tag{1.27}$$

In order to generate variates from a normal distribution with parameters μ and σ, we employ the central limit theorem [33]. The central limit theorem briefly states that if $x_1, x_2, \ldots, x_n$ are n independent random variates, each having the same probability distribution with $E(X_i) = \mu$ and $Var(X_i) = \sigma^2$, then the sum $\sum X_i = X_1 + X_2 + \cdots + X_n$ approaches a normal distribution as n becomes large. The mean and variance of this normal distribution are given by the following equations

$$E\left(\sum X_i\right) = n\mu \tag{1.28}$$

$$Var\left(\sum X_i\right) = n\sigma^2 \tag{1.29}$$

The procedure for generating normal variates requires k random numbers $r_1, r_2, \ldots, r_k$. Since each r_i is a uniformly distributed random number over the interval [0,1], we have that

$$E(r_i) = \frac{a+b}{2} = \frac{1}{2} \tag{1.30}$$

$$Var(r_i) = \frac{(b-a)^2}{12} = \frac{1}{12} \tag{1.31}$$

Using the Central Limit theorem the sum $\sum r_i$ of these k random numbers approaches the normal distribution as follows

$$\sum r_i \sim N\left(\frac{k}{2}, \frac{k}{\sqrt{12}}\right) \tag{1.32}$$

that is

$$\frac{\sum r_i - k/2}{k/\sqrt{12}} \sim N(0,1) \tag{1.33}$$

If we consider the normal distribution with parameters μ and σ from which we want to generate normal variates x then from eqs. 1.32 and 1.33 we have

$$\frac{x - \mu}{\sigma} = \frac{\sum r_i - k/2}{k/\sqrt{12}} \tag{1.34}$$

ISSN: 2279-8773

that became the following

$$x = \sigma\sqrt{\frac{12}{k}}\left(\sum r_i - \frac{k}{2}\right) + \mu. \tag{1.35}$$

This equation provides us with a simple formula for generating normal variates with a mean μ and standard deviation σ. The value of k has to be very large, became the larger it is the better is the accuracy.

Sampling from binomial distribution

In a binomial distribution [32] let assume that p is be the probability of success and $q = 1 - p$ the probability of a failure. If X is a random variable indicating the number of successes in n trials, it will follow the Binomial distribution. The probability density function of X is represented by the following equation

$$p(k) = \binom{n}{k} p^k \; q^{n-k}, \;\; k = 0, 1, 2, \;\ldots \tag{1.36}$$

where k is the number of successes in n trials. The expectation and variance of the binomial distribution are:

$$E(X) = np \tag{1.37}$$

and

$$Var(X) = npq \tag{1.38}$$

We can generate variates from a binomial distribution with a given p and generating n random numbers (after setting a variable k_0 equal to zero). For each random number $r_i, i = 1, 2, \ldots, n$, a check is made and the variable k_i is incremented as follows:

$$k_i = \begin{cases} k_{i-1} + 1 & \text{if } r_i < p \\ k_{i-1} & \text{if } r_i > p \end{cases}$$

The final quantity k_n is the binomial variate.

Sampling from a Poisson distribution

The Poisson distribution [32] models the occurrence of a particular event over a time period. Let λ be the average number of occurrences during a unit time period, then the number of occurrence x during a unit period has the following probability density function

$$p(n) = e^{-\lambda}(\lambda^n/n!), n = 0, 1, 2, \ldots \tag{1.39}$$

ISSN: 2279-8773

It can be demonstrated [34] that the time elapsing between two successive occurrences of the event is exponentially distributed with mean $1/\lambda$, i.e., $f(t) = \lambda e - \lambda t$. One method for generating Poisson variates involves the generation of exponentially distributed time intervals $t_1, t_2, t_3, \ldots$ with an expected value equal to $1/\lambda$. These intervals are accumulated until they exceed 1, the unit time period

$$\sum_{i=1}^{n} t_i < 1 < \sum_{i=1}^{n+1} t_i \tag{1.40}$$

The stochastic variate n is simply the number of events occurred during a unit time period. Now, since $t_i = \frac{1}{-\lambda} log\ r_i$, n can be obtained by simply summing up random numbers until the sum for $n + 1$ exceeds the quantity $e^{-\lambda}$ as evidenced in the equation

$$\sum_{i=1}^{n} r_i > e{-}\lambda > \sum_{i=1}^{n+1} r_i. \tag{1.41}$$

1.4 Model validation

Simulation models are increasingly being used to solve problems and to aid in decision-making. Starting from this assumption, a model should be developed for a specific purpose (or application) and its validity determined with respect to that purpose. If the purpose of a model is to answer a variety of questions, the validity of the model needs to be determined with respect to each question. Numerous sets of experimental conditions are usually required to define the domain of a model's intended applicability. In fact, a model may be valid for one set of experimental conditions but not valid in another. Moreover, a model is considered valid for a set of experimental conditions if its accuracy is within an acceptable range defined by considering the overall aspects that the model is able to satisfy.

This usually requires that the output variables of interest (i.e., the model variables used in answering the questions that the model is being developed to answer) have to be identified and their required amount of accuracy be specified. The model validation is a non costless practice. In fact, it can be quite significant, especially when extremely high confidence is required to the model. In Figure 1.6 a typical trend of the costs in function of the model confidence is reported.

ISSN: 2279-8773

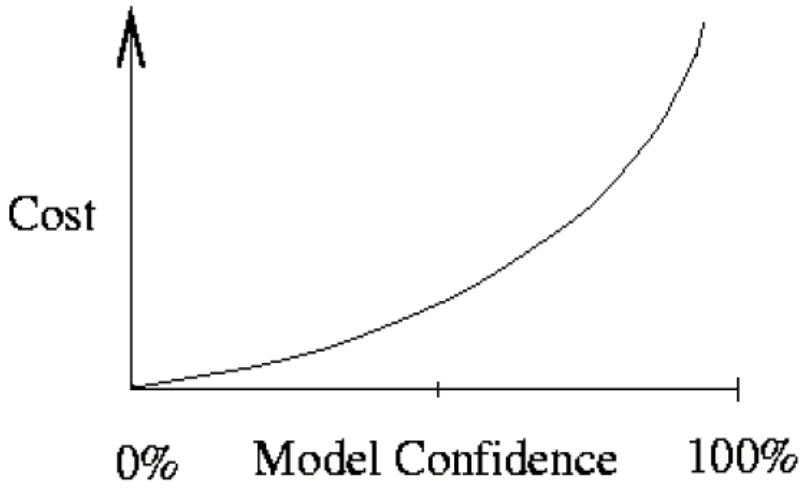

Figure 1.6: Representation of the costs of a model respect to its confidence. The graph show the exponential evolution of costs approaching the 100% of confidence.

1.4.1 Basic concepts in validation

There are four basic approaches for deciding whether a simulation model is valid. Each of the approaches requires the model development team to conduct verification and validation as part of the model development process.

The first approach, the most frequently used one, is for the model development team itself to make the decision as to whether a simulation model is valid. A subjective decision is made based on the results of the various tests and evaluations conducted as part of the model development process.

However, as this approach is extremely dependent by the development team, it is usually better to use one of the two approaches, depending on which situation applies.

In the second approach the user(s) of the model is heavily involved with the model development team in deciding the validity of the simulation model. In this approach the focus of determining the validity of the simulation model moves from the model developers to the model users aiding in model credibility.

The third approach, usually called Independent Verification and Validation (IV&V) [35], uses a third independent party to decide whether the simulation model is valid. The third party is independent from both the simulation development team(s) and the model user(s). The IV&V approach is mainly used when developing large-scale simulation models, whose develop-

ISSN: 2279-8773

ments usually involve several teams. This approach is also used to increase the model credibility, especially when the problem tackled by the simulation model is associated with high costs. In this approach the third party must have a deeper understanding of the intended purpose(s) of the simulation model in order to lead the group in their choices.

The last approach for determining whether a model is valid is to use a scoring model [36, 37]. Scores (or weights) are determined subjectively when conducting various aspects of the validation process and then combined to determine category scores and an overall score for the simulation model. A simulation model is considered valid if its overall and category scores are greater than some passing score(s). This approach is seldom used in practice because a model may receive a passing score and yet have a defect that have to be corrected, the passing scores must be decided in a subjective way, the score(s) may cause over confidence in a model arguing that one model is better than another.

1.4.2 Validation techniques

Several techniques [38] used to verify and validate the submodels and the overall model have been developed and tested in the years. The most used are reported and described in this section.

Animation: The operational behavior of the model is displayed graphically as the model moves through time. For example the movements of parts through a factory during a simulation run are shown graphically.

Comparison to Other Models: Various results of the simulation model being validated are compared to results of other (possible valid) models. As an example, simple cases of a simulation model are compared to known results of analytic models, and the simulation model is compared to other simulation models that have been validated.

Degenerate Tests: The degeneracy of the behavior of a model is tested by appropriate selection of values of the input and internal parameters.

Event Validity: The events of occurrences of the simulation model are compared to those of the real system to determine if they are similar. For example, compare the number of fires in a fire department simulation.

Extreme Condition Tests: The model structure and outputs should be plausible for any extreme and unlikely combination of levels of factors in the system. For example, if in-process inventories are zero, production output should usually be zero.

ISSN: 2279-8773

Face Validity: Asking individuals knowledgeable about the system whether the model and/or its behavior are reasonable. For example, is the logic in the conceptual model correct and are the model's input-output relationships reasonable?

Internal Validity: Several runs of a stochastic model are made to determine the amount of stochastic variability in the model. A large amount of variability (lack of consistency) may cause the model results to be questionable and if this came from the problem entity, appropriateness of the policy or system being investigated may be questionable.

Operational Graphics: Values of various performance measures (e.g., the number in queue and percentage of servers busy) are shown graphically as the model runs. In this way the dynamical behaviors of performance indicators are visually displayed as the simulation model runs through time to ensure they behave correctly.

Parameter Variability - Sensitivity Analysis: This technique consists of changing the values of the input and internal parameters of a model to determine the effect upon the model behavior or output. The same relationships should occur in the model as in the real system. This technique can be used both qualitatively (directions only of outputs) and quantitatively (both directions and (precise) magnitudes of outputs). Those parameters that are sensitive, (i.e., cause significant changes in the model behavior or output) should be made sufficiently accurate prior to using the model.

Predictive Validation: The model is used to predict (forecast) the system behavior that is then compared with the model forecast to determine possible incongruences. The system data may come from an operational system or be obtained by conducting experiments on the system itself.

Traces: The behavior of different types of specific entities in the model are traced (followed) through the model to determine if its logic is correct and if the necessary accuracy is obtained.

Turing Tests: Individuals who are knowledgeable about the operations of the system being modeled are asked if they can discriminate between system and model outputs [39]. (Schruben 1980)

1.4.3 Overview on modeling errors

Assuming that the parametrized model to be validated takes the form,

ISSN: 2279-8773

$$x_m(t) = f_m(x_m(t),\ \theta_m,\ u(t),\ t) \tag{1.42}$$

where $x_m(t) \in \mathbb{R}^n$ is the state variable vector of the model, $u(t) \in \mathbb{R}^p$ is the input vector, and θ_m is the model parameter vector, which is known. On the basis of this model, the real behavior of the system can generally be represented as,

$$x_r(t) = f_m(x_r(t),\ \theta_m,\ u(t),\ t) + e_m(t) \tag{1.43}$$

where $x_r(t) \in \mathbb{R}^n$ is the state vector of the system, $e_m(t) \in \mathbb{R}^n$ is the modeling error vector. It is assumed in equation 1.43 that the real system has the same number of state variables as the model. This representation does not limit the generality of the representation since the errors introduced by erroneous state aggregations in deriving model (eq. 1.42) can also be represented by the error term $e_m(t)$. In order to make the modeling error identication possible, an appropriate representation of the error term $e_m(t)$ in equation 1.43 is required. This representation should be obtained by making use of the a priori knowledge about the possible modeling errors. Basically, modeling errors may be introduced in each stage of the modeling process.

The *a priori* knowledge concerning the modeling errors can be obtained through the analysis of the modeling process and the model itself making use of mathematical representation that allows the identification of the modeling errors by comparing the outputs coming from the observed system with data produced by simulation of the erroneous model.

Improperly defined Experimental Frame

In defining the boundaries of the process or system to be modelled, some important components may be missed, some significant disturbances to the system may be improperly neglected and so on. All of these aspects introduce errors into the model. The Experimental Frame is the formalization of the experimental conditions (inputs applied to the system, outputs observed, criteria of acceptance, ...) and as such the above mentioned modeling errors can be formally expressed as Experimental Frame errors [40].

ISSN: 2279-8773

Improperly characterized Model Structure

Due to lack of knowledge of the mechanism of the process to be modelled, or due to an oversimplification of the model, a wrong model structure can be assumed. Typical errors include choosing an incorrect number of state variables or incorrectly assuming non-linear behavior. Structural errors may accidentally be produced through incorrect choice of parameters (usually, 0), whereby some part of the model structure vanishes, thereby altering the model structure.

Inaccurate estimates of the Model Parameters

Either by improper or inadequate data used for parameter estimation or by ill designed estimation algorithms, incorrect parameter values may be used.

Usually, each modeling error affects only a subspace of the n-dimensional state space, and can hence be represented in equation 1.43 with a term $F_i d_i(t)$, where $F_i \in R^{nxs_i}$ and $d_i \in R^{s_i}$. The vectors of F_i span the subspace affected by the concerned modeling error. Whereas F_i is called the feature vector or feature matrix of the modeling error, $d_i(t)$ represents the magnitude of the modelling error, and is generally unknown and time-varying. Thus, equation 1.43 can be rewritten as,

$$x_r(t) = f_m(x_r(t),\ \theta_m,\ u(t),\ t) + \sum_{i=0}^{n} F_i d_i(t) \tag{1.44}$$

Since it is usually not possible to predict all possible modeling errors, it is necessary to include a special feature matrix (F_0) in equation 1.44 to represent modeling errors which were not explicitly modelled. Obviously, the n-dimensional identity matrix is suitable for that purpose. To allow for meaningful error identification, some assumptions are made with respect to equation 1.44:

- The individual errors are written in ?additive? form:

$$\nu_r = \nu + \delta\nu \tag{1.45}$$

  Such a choice of individual error terms is always possible without loss of generality.

- Simultaneously occurring errors are assumed to be either additive, or sufficiently small to allow for an linear approximation:

$$f(A+\delta A, B+\delta B) \cong f(A,B) + \frac{\partial f}{\partial A}(A,B)\delta A + \frac{\partial f}{\partial B}(A,B)\delta B \tag{1.46}$$

with A and B different errors. Though such an assumption, nonlinear effects can always be lumped into an extra error term using the F_0 special feature matrix mentioned above.

1.5 Multiscale Modeling of complex systems

Simulation based on multiscale modeling and computation is a rapidly evolving area of research that is having a fundamental impact on computational science and applied mathematics and will influence as well the way we view the relationship between mathematics and science. Even though multiscale problems have been studied in mathematics since long, the current excitement is driven mainly by the use of mathematical models in the applied sciences like material science, chemistry, fluid dynamics, and biology. Problems in these areas are often multiphysics in nature. Namely, the processes occurring at different scales are governed by physical laws having a different character. For example, quantum mechanics at one scale and classical or fluid mechanics at another. Emerging from this intense activity is a need for new mathematics and new ways of interacting with mathematics. Fields such as mathematical physics and stochastic processes, which have so far remained in the background as far as modeling and computation of complex systems starting from first principles is concerned, will move to the front. New questions will arise, new priorities will be set as a result of the rapid evolution in the computational fields.

There are several reasons for the timing of the current interest. Modeling at the level of a single scale, such as molecular dynamics or continuum theory, is becoming relatively mature. Our computational capability has reached the stage in which serious multiscale problems can be considered (and there is an urgent need from science and technology, nano-science being a good example) for multiscale modeling techniques.

ISSN: 2279-8773

1.5.1 Approaches to multiscaling

A traditional multiscale approach is a Fourier analysis that has since long been used as a way of representing functions according to their components at different scales. More recently, this multiscale, multiresolution representation has been made much more efficient through wavelets. On the computational side, several important classes of numerical methods have been developed which address explicitly the multiscale nature of the solutions. These include multigrid methods, domain decomposition methods, fast multipole methods, adaptive mesh refinement techniques, and multiresolution methods using wavelets. From a modern perspective, the computational techniques described above are aimed at efficient representation or solution of the fine-scale problem. For many practical problems, full representation or solution of the fine-scale problem is simply impossible for the foreseeable future because of the overwhelming computational costs. Therefore we must seek more efficient alternative approaches.

Protein folding is a typical example of multiple time scales. While the time scale for the vibration of the covalent bonds is on the order of femtoseconds (10^{-15} s), folding time for the proteins may very well be on the order of seconds. Well-known examples of problems with multiple length scales include turbulent flows, mass distribution in the universe, and vortical structures on the weather map [41]. At the same time different physical laws may be required to describe the system at different scales. Take the example of fluids. At the macroscale (meters or millimeters), fluids are accurately described by the density, velocity, and temperature fields, which obey the continuum Navier-Stokes equations. On the scale of the mean free path, it is necessary to use kinetic theory (Boltzmann's equation) to get a more detailed description in terms of the one-particle phase-space distribution function. At the nanometer scale, molecular dynamics in the form of Newton's law has to be used to give the actual position and velocity of each individual atom that makes up the fluid. If a liquid such as water is used as solvent for protein folding, then the electronic structures of the water molecules become important, and these are described by Schrödinger's equation in quantum mechanics. The boundaries between different levels of theories may vary, depending on the system being studied, but the overall trend described above is generally valid. At each finer scale, a more detailed theory has to be used, giving rise to more detailed information on the system.

ISSN: 2279–8773

1.5.2 Analytic techniques

Analytic techniques can be used as a classical approach to derive effective models at the scale of interest. As an example of the application of such a technique let us consider the averaging method in which the multiscale nature of the problem can be formulated as a system of ordinary differential equations in the action-angle variables by writing it as follows

$$\varphi_t = \frac{1}{\varepsilon}\omega(I) + f(\varphi, I)I_t = g(\varphi, I)$$

where φ is the fast variable, which varies on the time scale of $O(\varepsilon)$, while $\varepsilon \ll 1$; I is the slow variable, which mainly varies on the time scale of $O(1)$ (f and g are assumed to be 2π-periodic in φ).

Another example of a mathematical technique for approaching multiscale problems is the homogenization method for which we can consider the problem

$$\frac{\partial u^\varepsilon}{\partial t} = \nabla \cdot (a(x, \frac{x}{\varepsilon})\nabla u^\varepsilon(x,t)), \quad x = \Omega \tag{1.47}$$

with the boundary condition $u^\varepsilon\|_{\partial\Omega} = 0$. In this problem the multiscale nature comes from the coefficients $a(x, \frac{x}{\varepsilon})$, which contain two scales: a scale of $O(\varepsilon)$ and a scale of $O(1)$. Not only is (1.47) a nice model problem for the homogenization technique, it also describes important physical processes such as heat conduction in a composite material (for simplicity let us assume that $a(x, y)$ is periodic in y). It can be shown [42] that for $\varepsilon \ll 1$, $u^\varepsilon(x,t)$ can be expressed in the form

$$u^\varepsilon(x,t) = U(x,t) + \varepsilon u_1(x, \frac{x}{\varepsilon}, t) + \varepsilon^2 u_2((x, \frac{x}{\varepsilon}, t)) + \ldots, \tag{1.48}$$

where U satisfies a homogenized equation

$$\frac{\partial U}{\partial t} = \nabla \cdot (A(x)\nabla U(x,t)). \tag{1.49}$$

Here $A(x)$ may be thought of as being the effective coefficient describing the effective properties of the system on the scale of $O(1)$. Determining $A(x)$ usually requires solving families of the so called cell problems.

ISSN: 2279-8773

In the one dimensional case, however, $A(x)$ is simply given by the harmonic average

$$A(x) = (\int_0^1 \frac{1}{a(x,y)} dy)^{-1}. \tag{1.50}$$

1.5.3 Empirical techniques

Many other mathematical approaches have been developed to study multiscale problems, including boundary-layer analysis [43], semiclassical methods [44], geometric theory of diffractions [45], stochastic mode elimination [46], and renormalization group methods [47], [48]. Despite this progress, purely analytical techniques are still very limited when it comes to problems of practical interest. As a result, the overwhelming majority of problems have been approached using empirical techniques to model the small scales in terms of the macroscale variables using empirically derived formule. As a matter of fact, a large part of the progress in physical sciences lies in such empirical modeling. A familiar example is the case of the continuum theory of fluid dynamics. To derive the system of equations for fluids, we apply Newton's law to an arbitrary volume of fluid denoted by Ω:

$$\frac{D}{Dt} \int_\Omega \rho u dV = F(\Omega) \tag{1.51}$$

where $\frac{D}{Dt}$ is the material derivative, ρ and u are the density and velocity fields respectively, and $F(\Omega)$ is the total force acting on the volume of fluid in Ω. The forces consist of body forces such as gravity, which we neglect for the present argument, due to the long-range interaction of the molecules that make up the fluid, and forces due to the macroscopic-range interaction between the molecules, such as the Van der Waals interaction. In the continuum theory, the short-range forces are represented as a surface integral of the stress tensor τ, which is a macroscopic idealization of the small scale effects,

$$F(\Omega) = \int_{\partial\Omega} (\tau \hat{n}) ds \tag{1.52}$$

where $\hat{n}$ is the unit outward normal of Ω. The stress τ can be expressed as $\tau = -pI + \tau_d$, where p is the pressure, I is the identity tensor, and τ_d is the dissipative part of the stress. In order to close the system, we need

to express τ_d in terms of u. In the simplest empirical approximation, τ_d is assumed to be a linear function of ∇u. This leads to

$$\tau_d = \mu \frac{\nabla u + (\nabla u)^T}{2} \tag{1.53}$$

where μ is called the viscosity of the fluid. Substituting this into Newton's law and adding the incompressibility condition gives rise to the well known Navier-Stokes equation:

$$\rho(u_t + (u \cdot \nabla)u) + \nabla p = \mu \Delta u \qquad \nabla \cdot u = 0 \tag{1.54}$$

In such a macroscopic description, all molecular details of the liquid are lumped into a single parameter, the viscosity. Fluids for which Eq. 1.53 gives an accurate description of the small-scale effects are called Newtonian fluids. This simple derivation illustrates how, in general, continuum models in the form of partial differential equations are derived. One typically starts with some universal conservation laws such as Eq. 1.51. This requires introducing certain currents or flux densities, which are then expressed by some postulated constitutive relationships such as 1.53. In this way, we obtain the heat equation for thermal conduction by postulating Fourier's law, the diffusion equation for mass transport using Fick's law, and the porous medium equation using Darcy's law. Such empirical ad hoc descriptions of the small scales are used almost everywhere in science and engineering. In molecular dynamics, empirical potentials are used to model the forces between atoms, mediated by the electrons. In kinetic theory, empirical collision kernels are used to describe probabilistically the short-range interaction between the atoms and the molecules. Other examples include plasticity, crack propagation, and chemical reactions. While much progress has been made using such empirical approaches, their shortcomings have also been recognized, especially in recent years, since numerical simulations based on the empirical models are now accurate enough that the modeling error can be clearly identified. Microscale simulation methods such as electronic structure calculations have matured, enabling us to ask more ambitious questions. Moreover, the empirical approach often lacks information about how microstructural changes (such as the conformation of polymers in a polymeric fluid) affect the macroscale properties of the system.

ISSN: 2279–8773

1.5.4 Examples of Multiscale Problems

In view of the limitations of the empirical approach, several "first principle"-based multiscale methods have been proposed in recent years. Some of these traditional and modern computational multiscale techniques are given in Table 1.2.

Table 1.2: Traditional and modern computational multiscale techniques. Traditional multiscale techniques focus on resolving the fine-scale problem. Modern multiscale techniques try to reduce the computational complexity by using special features in the fine-scale problem, such as scale separation.

Traditional Techniques	Recent Techniques
Multigrid Method	Car-Parrinello Method
Domain Decomposition	Quasi-Continuum Method
Multiresolution Methods	Superparametrization
Adaptive Mesh Refinement	Heterogeneous Multiscale Method
Fast Multipole Method	Vanden-Eijnden's Method
Conjugate Gradient Method	Coarse-Grained Monte Carlo Models
	Adaptive Model Refinement
	Patch Dynamics

Quasicontinuum Method

In the continuum theory of nonlinear elasticity, we are often interested in finding the displacement field by solving a variational problem

$$\min_u E(u) = \int_\Omega f(\nabla u) dx \tag{1.55}$$

where E is the total elastic energy, u is the displacement field, and f is the stored energy functional, subject to certain loading or boundary conditions. This approach takes for granted that the function f is explicitly given. Actually the process of finding f is rather empirical and often even crude. A different methodology called the quasicontinuum method has been proposed in Refs. [49] and [50] for the analysis of crystalline materials. In this case the microscopic model comprises molecular mechanics of the atoms that make up the crystal. Given a macroscopic triangulation of the material, let V_H be the standard continuous piecewise-linear finite-element space over this triangulation.

ISSN: 2279–8773

For $U \in V_H$, ∇U is constant on each element K. Let $E_K(U)$ be the energy of a unit cell in an infinite volume uniformly deformed according to the constant deformation gradient $\nabla U \mid_K$. In the quasicontinuum approximation, the total energy associated with the trial function U is then given by

$$\tilde{E}(U) \sum_K n_K E_K(U) \tag{1.56}$$

where n_K is the number of unit cells in the element K. This approach bypasses the necessity of modeling f empirically. Instead, the effective f is computed on the fly using microscopic models. What we have described is the simplest version of the quasicontinuum method. There are many improvements, in particular to deal with defects in the crystal [50].

Kinetic-Hydrodynamic Models of Complex Fluids

Consider, for example, polymers in a solvent. The basic equations follow again from that of mass and momentum conservation:

$$\rho(u_t + (u \cdot \nabla)u) + \nabla p = \mu_s \Delta u + \nabla \cdot \tau_p \quad \nabla \cdot u = 0 \tag{1.57}$$

Here we have decomposed the total stress into two parts: one part, τ_p, due to the polymer and the other part due to the solvent, for which we used Newtonian approximation; μ_s is the solvent viscosity. Traditionally, τ_p is modeled empirically using constitutive relations. The most common models are a generalized Newtonian model and various viscoelastic models. It is generally acknowledged that it is an extremely difficult task to construct such empirical models in order to describe the flow under all experimental conditions. An alternative approach was proposed in the classical work of Kramers, Kuhn, Rouse et al. [51]. Instead of using empirical constitutive relationships, this approach makes use of a simplified kinetic description for the conformation of the polymers. In the simplest situation, the polymers are assumed to be dumbbells, each of which consists of two beads connected by a spring. Its conformation is therefore described by that of the spring. The dumbbells are convected and stretched by the fluid, and at the same time they experience spring and Brownian forces:

$$\gamma(Q_t + (u \cdot \nabla)Q - (\nabla u)^T Q) = F(Q) + \sqrt{k_B T \gamma}\dot{W}(t) \tag{1.58}$$

Here Q denotes the conformation of the dumbbell, $F(Q)$ is the spring force, γ is the friction coefficient, $\dot{W}(t)$ is temporal white noise, k_B is

the Boltzmann constant, T is the temperature, and I is the identity tensor. If we have Q, we can compute the polymer stress τ_p via

$$\tau_p = nk_BTI + \mathbb{E}(F(Q) \bigotimes Q), \tag{1.59}$$

where n is the polymer density and $\mathbb{E}$ denotes expectation over the Brownian forces. These equations are valid in the dilute regime when direct interaction between polymers can be neglected. The dumbbell model is a very simplified one and in many cases needs to be improved. This can be done in a number of ways (see Ref [51]). Many other multiscale methods which are similar to those mentioned above have been developed in the past few years. We mention in particular the work of Abraham et al. on coupling finite element continuum analysis with molecular dynamics and tight binding [52], the work on coupling kinetic equations with hydrodynamic equations, Vanden-Eijnden's method for solving stochastic ordinary differential equations with multiple time scales [53], superparametrization techniques in meteorology in which the parameters for turbulent transport are determined dynamically by local microscale simulations, and the work of Kevrekidis et al. on bifurcation analysis based on microscopic models [54]. By explicitly taking advantage of the separation of scales, these methods become much more efficient than solving the full fine-scale problem. This is a common feature of the new class of multiscale methods we are interested in. In contrast, traditional multiscale techniques such as the original multigrid methods are rather blind to the special features of the problem, since they are aimed at solving the full fine-scale problem everywhere in the macroscale domain. Of course many practical problems such as turbulent flows do not have separation of scales. For these problems, other special features, such as selfsimilarity in scales, must be identified first before we have a way of modeling them more efficiently than simply solving the fine-scale problem by brute force or resorting to ad hoc models.

Molecular Dynamics

Usual MD approaches to the evaluation of the properties of large molecular systems are based on the integration of classical mechanics equations.

This makes Molecular Dynamics a deterministic technique: given an initial set of positions and velocities, the subsequent time evolution

ISSN: 2279–8773

is *in principle*[1] completely determined. The computer calculates a trajectory of the system in a $6N$-dimensional phase space ($3N$ positions and $3N$ momenta).

However, the classical mechanics approach of MD has intrinsic limits which can be singled out in a simple way by defining the de Broglie thermal wavelength [55]:

$$\Lambda = \sqrt{\frac{2\pi\hbar}{M\kappa_B T}} \tag{1.60}$$

where M is the mass of the system. The classical approximation is valid when $\Lambda \ll a$, where a is the average interparticle distance. For a gas this quantity is approximately $(V/N)^{1/2}$ where V is the volume and N is the number of particles. When the thermal de Broglie wavelength is much smaller than the interparticle distance, the gas can be considered to be a classical or Maxwell-Boltzmann gas. On the other hand, when the thermal de Broglie wavelength is of the order of (or larger than) the interparticle distance, quantum effects will dominate and the gas must be treated as a Fermi gas or a Bose gas, depending on the nature of the gas particles. The critical temperature is the transition point between these two regimes. At the critical temperature, the thermal wavelength is approximately equal to the interparticle distance. That is, the quantum nature of the gas will become evident for:

$$\frac{V}{N\Lambda^3} \leq 1 \tag{1.61}$$

and in this case the gas will obey Bose-Einstein statistics or Fermi-Dirac statistics, whichever is appropriate. On the other hand, for

$$\frac{V}{N\Lambda^3} \gg 1 \tag{1.62}$$

the gas will obey Maxwell-Boltzmann statistics. The classical approximation is poor for light systems and when T is sufficiently low. Molecular dynamics results should be interpreted with caution in these regions. There are however other intrinsic limits in molecular dynamics simulations. In fact in MD atoms interact with each other and these interactions originate forces which act upon atoms, and atoms move under the action of these instantaneous forces. As the atoms move, their

[1] in practice, the finiteness of the integration time step and of the electronic representation of numbers (in other words the arithmetic rounding) might cause the computed trajectory deviate from the true one.

ISSN: 2279-8773

relative positions change and forces change as well. Forces are usually obtained as gradient of the potential energy function and especially depend on the particle positions. The accuracy of the simulation therefore is very sensitive to the suitability of the potential chosen to describe the interaction of the components of the system. MD simulations are performed on systems typically containing thousands, millions or even billions of atom. Simulation times on their side can range, under appropriate conditions, from picoseconds to microseconds. A simulation is "safe" with respect to its minimum duration when duration in time is much longer than the relaxation time of the quantities of interest. However, different properties may have different relaxation times. In particular, systems tend to become slow and sluggish in the proximity of phase transitions, and it is not uncommon to find cases where the relaxation time of a physical property is orders of magnitude larger than times achievable by computer simulations. On the contrary, there is a problem of "safety" also with respect to its maximum duration. In fact, the error accumulation at each time step may sum up to an amount comparable with the integration variables when the number of steps is exceedingly large. The size of the system can also constitute a problem. For example in the case in which one has to compare the size of the MD cell with the correlation lengths of the spatial correlation functions of interest there may be problems when the size of the cell is too small. Correlation lengths, in fact, may increase or even diverge in proximity of phase transitions. Therefore, the result may become no longer reliable when the correlation length becomes comparable with the box length. This problem can be partially alleviated by a method known as finite size scaling. This consist of computing a physical property A using several boxes having different sizes L, and then fitting the results using the relationship:

$$A(L) = A_\circ + \frac{c}{L^n} \tag{1.63}$$

with $A_\circ$, c and n being the best fit parameters. $A_\circ$ then corresponds to $\lim_{L\to\infty} A(L)$, and should therefore be taken as the best estimate for the physical quantity. Despite the above discussed limitations most of the observable can be satisfactorily evaluated in multiscale treatments [56, 57].

ISSN: 2279-8773

Chapter 2

Concurrent computing

2.1 Introduction

The European Union has since long promoted the assemblage of a cluster of large scale supercomputer platforms in order to tackle the solution of the mentioned Grand Challenges in computational sciences. At present, however, the evolution of networking and computer technologies allows to go beyond the policy of individual large computer centers machines in favour of clusters of out-of-the-shelf PCs. Moreover at the same time Computational applications are increasingly becoming an assemblage of different packages needing the converged effort of different expertises and multiscale modeling. This leverages on the present effort at creating worldwide Grid platforms and fostering the development and implementation on the Grid of ICT applications in all fields of human activities.

For applications concerning the lowest level usage of the Grid platform (data transfer) a broad bandwidth, a safe and secure transfer protocol and an authenticated access are needed. A distributed usage of the information on the Grid dramatically requires new services mainly consisting of intelligent tools for information representation and handling. Moreover, the shared and distributed usage of knowledge is, in absolute, one of the most complex tasks. In fact, it implies the coordination of the expertise, the cross fertilization of the know how, the protection of the intellectual property and the stimulation of teamed innovative research. Additional specific difficulties arise when this has to be designed and implemented on a highly distributed and heterogeneous context and the subject to deal with is the interpretation of

ISSN: 2279-8773

transformations in matter. In this context, the Grand Challenge is the realistic *a priori* simulation of the multiscale type, starting from the microscopic level of the molecular interaction and, therefore, ranging from molecular to human level. Inevitably such simulation is necessarily founded on competences and skills scattered worldwide as typical of Grids.

In this chapter, the evolution of computer technologies and platforms towards distributed computing and its impact on the way a Grid empowered Molecular Simulator for materials science applications has been built, is described.

2.2 Concurrency on a single processor

The single-processor architectures are based on the well known Von Neumann model, sketched in Fig. 2.1. In this model the processing unit first fetches an instruction from the memory that subsequently is interpreted and executed after the fetching of the operands involved. Therefore, in a single-cycle the CPU performs the following operations:

- ***Fetch the next instruction***: the next instruction referenced by the program counter is transferred from memory to CPU;
- ***Decode***: the fetched instruction is interpreted and related circuitry activated;
- ***Fetch the operands***: the referenced operands are transferred from memory and stored in the registers;
- ***Execute***: the decoded instruction is executed;
- ***Check for interrupt***: the processor checks for signal of interrupt issued by the operating system or any other process and decides accordingly to stop or to solve the interrupt and continue;
- ***Store the results***: the results are transferred to the appropriate memory locations;
- ***Increment the counter*** the counter is updated to allow the fetch of the next iteration.

The Von Neumann architecture has represented the first step in the process of building an efficient automatic handling of information. Year by year, the original architecture has been modified to increase its speed

ISSN: 2279-8773

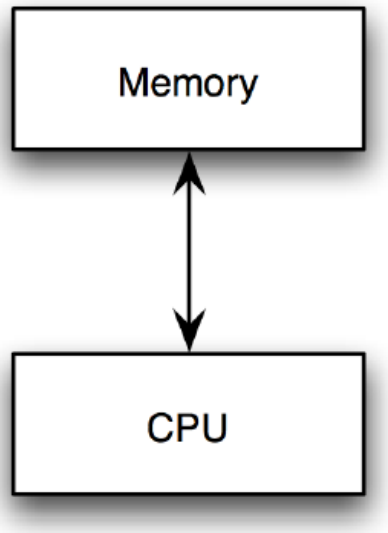

Figure 2.1: The Von Neumann model.

and improve its efficiency. The progress towards faster computers has been largely based, in the past, on technology advances in producing increasingly more integrated circuitry. This has made the CPU more efficient, the memories more capable, the buses more transfer-performing.

The ability of further integrating an increasing number of circuitry elements on a chip has led to a silicon design integrating into the same processor socket multiple execution cores (multicore processors).

Despite the fact that the multi-core processor plugs multiple "execution cores" directly into a single processor socket, the operating system perceives each of them as a discrete logical processor with all the associated execution resources. The idea behind this implementation is the strategy of *divide et impera*. This means that by dividing the computational work traditionally performed by the single processor's core in traditional processors and spreading it over multiple execution cores, a multi-core processor can perform more work within a given clock cycle. To enable this improvement, the software running on the platform must be written in a way suited to spread the workload across multiple execution cores. This functionality is called thread-level parallelism or "threading". Applications and operating systems that are written to support threading are referred to as "threaded" or "multi-threaded". A processor equipped with thread-level parallelism can execute com-

ISSN: 2279-8773

pletely separate threads of code. This can mean one thread running from an application and a second thread running from an operating system, or parallel threads running from within a single application.

Circuital improvements have fuelled a constant advance in computing speed that has been quantified in the past by the Moore law: "the computing speed doubles every two years". A graphical illustration of the Moore law is given in Fig. 2.2 where the CPU transistor counts are plotted against dates of introduction suggesting that the exponential growth with transistor count doubling every two years.

However, due to the intrinsic limit of this process (a signal cannot be faster than light speed in the related medium), progress has involved other key aspects of the computer and has prompted innovative research in several relevant fields, including optical circuitry, molecular devices, quantum computers, etc.

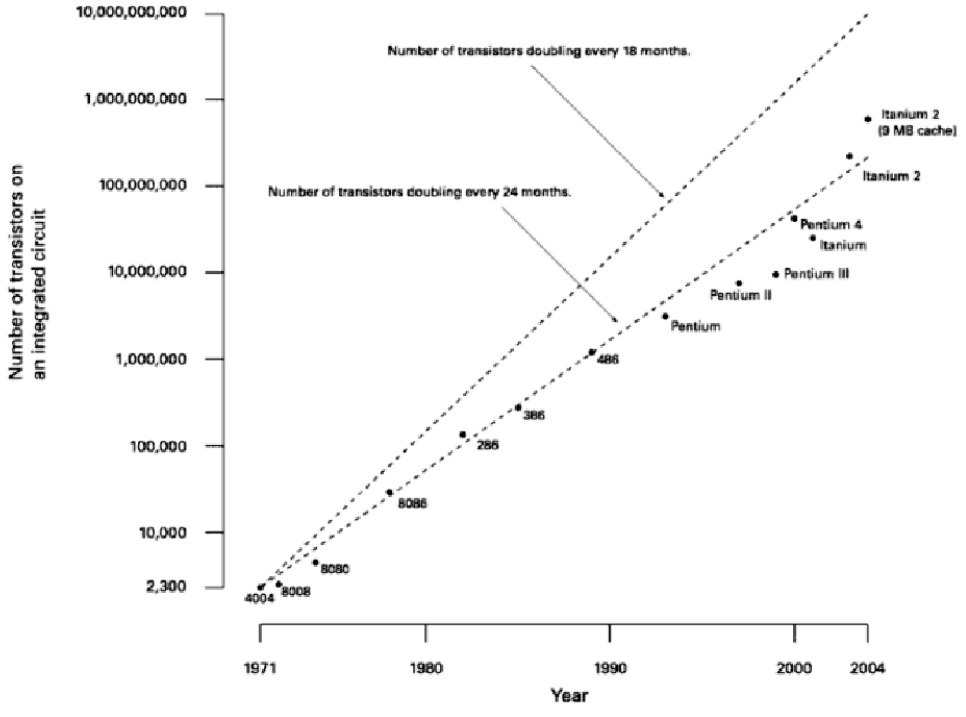

Figure 2.2: A graphical representation of the Moore's law.

ISSN: 2279–8773

2.2.1 Management concurrency

The first move towards data transfer concurrency is already built into the Von Neumann architecture that makes use of buses for parallel bit transfers. The concept of concurrent executing was also already proposed by Babbage and by Von Neumann for dealing with the various grid points of the differential archiectures. Another example of *ante litteram* parallelism is the first vacuum tube computer (ENIAC [58]) that was made of 25 independent computing units. However, most of the early days progress was obtained at management level. The concepts themselves of typing ahead, multiprogramming and time sharing did build up software and hardware concurrency in sequential (single CPUs) computers. Along the same line moved the use of dedicated (low level) CPUs for dealing, for example, with I/O internal communications.

The real progress, however, was reached by making in a single machine the circuitry extensively concurrent in several ways including CPU duplication, processing units segmentation, memory partitioning and very recently, as already mentioned, multi-core architecture. Such single machine concurrency is more frequently referred to as "*parallelism*".

2.2.2 Instruction level parallelism

When the parallelism is exploited at instruction level (Instruction Level Parallelism, **ILP**) the concurrent execution of a multiple flux of instructions is adopted in order to maximize the performance. This is obtained by making use of the pipelining as illustrated in Fig. 2.3. In a pipelined architecture the CPU is usually partitioned up into stages for each operation (including instruction decoding, arithmetic, and register fetching) with each stage processing one instruction at a time. In the case illustrated in Fig. 2.3 the following nomenclature is adopted:

- IF = Instruction Fetch
- ID = Instruction Decode
- EX = Execute
- MEM = Memory access
- WB = Register write back

In some cases such a multiple flux can be sorted out (by the compiler as well as by the hardware) on the basis of a partial sorting induced by

ISSN: 2279–8773

the logical dependencies of instructions, depending on the semantic of the program. The **ILP** paradigm was born in the seventies (together with superscalar architectures) by adopting a fine grain parallelism and evolving later in multiscalar and Very Long Instruction Word (VLIW) architectures.

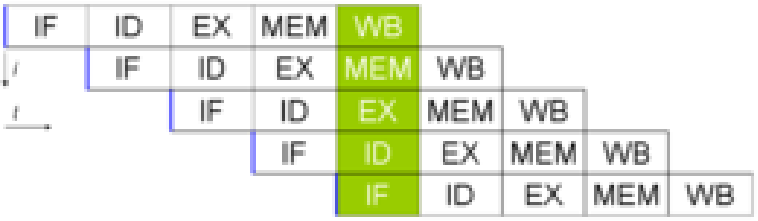

Figure 2.3: A graphical representation of the pipeline in a microprocessor.

Superscalar Architectures

In **ILP** architectures, even if instructions are pipelined, in each stage of the pipeline only one instruction can be executed. In superscalar architectures (see Fig. 2.4) the machine-cycle phases are concurrently executed by simultaneously dispatching multiple instructions to redundant functional units on the processor. Each functional unit is not a separate CPU core but an execution resource within a single CPU such as an arithmetic logic unit, a bit shifter, or a multiplier. The superscalar architecture is traditionally associated with several identifying characteristics applied within a given CPU core. Accordingly its main features are:

- Instructions are issued from a sequential instruction stream
- CPU hardware dynamically checks for data dependencies between instructions at run time (versus software checking at compile time)
- Multiple instructions per clock cycle are accepted.

All executing functional units, due to the fact that each of them is implemented to perform a well established class of machine operations, are totally dedicated. The involved operations include floating/fixed-point operations as well as load/store operations (the last class involves data movements from/to the main memory). The degree of parallelism of these architectures is confined between three and six; processors

belonging to this class are, for example, the MIPS 15000, DEC 21164, IBM Power4, ULtra Sparc III.

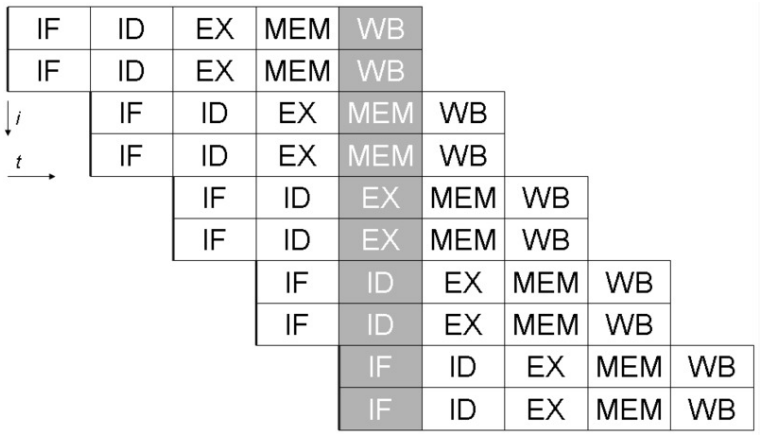

Figure 2.4: Superscalar architecture with double pipeline.

Multiscalar architectures

The most recent **ILP** paradigm is the Multiscalar [59]. In Multiscalar **ILP** architectures the granularity of the parallelism, exploited at instruction level, is greater than that of superscalar architectures. In these architectures the program loaded in the main memory is partitioned into many independent (in terms of logic) tasks which are distributed to the functional units, where the cycle-machine phases are applied to the instructions of the assigned task.

VLIW architectures

The VLIW (Very Long Instruction Word) architecture (see Fig 2.5) takes advantage of the capability of the compiler to compact independent operations in a larger single instruction word and to execute it on different functional units (after segmentation). The VLIW processor executes operation in parallel based on a fixed schedule determined when programs are compiled. Since determining the order of execution of operations (including which operations can execute simultaneously) is handled by the compiler, this means that the processor does not need scheduling hardware. As a result, VLIW CPUs offer significant computational power with less hardware complexity (but greater compiler

complexity) than is associated with most superscalar CPUs. Thus, it requires the adoption of refined compilers able to implement advanced techniques, such as Software pipelining and trace scheduling [60].

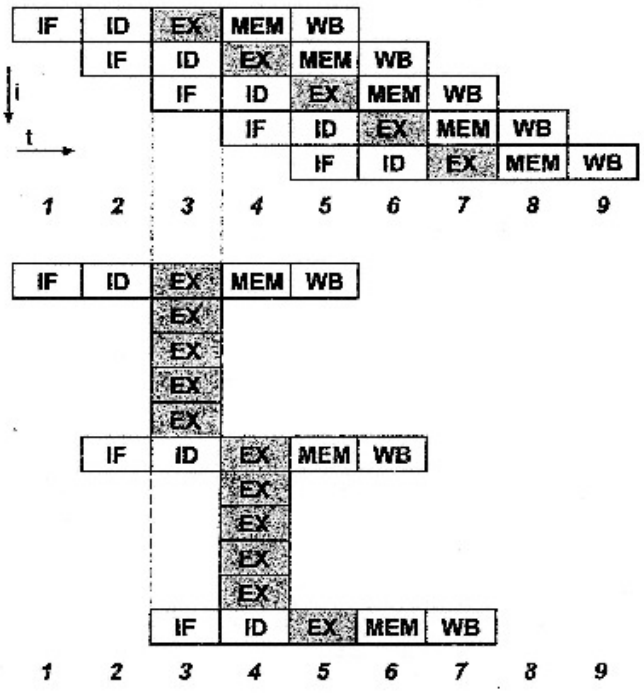

Figure 2.5: VLIW architecture.

2.2.3 Data Level Parallelism

The parallelism exploited at data level (data level parallelism, **DLP**) differs from the **ILP** in the granularity of the operands involved in the operations. Arithmetic operations are executed on data item arrays: in this way the paradigm becomes useful when one has to deal with applications involving a large number of vector and matrix elements operations.

Vector processors

Vector computers have been the first data parallel high performance computing systems. It has also been proposed to add a vector functional unit [61] to a superscalar processor. In a vector machine the

flux of instructions passes through the Scalar Control Unit that takes care of submitting the instruction, if of the vector type, to the Vector Control Unit. Such an instruction is executed by a functional pipeline. In Fig.2.6 a register-register vector architecture is sketched. The vector operands as well as the vector results are stored in vector registers having a fixed (as in the CRAY computers) or dynamic (as in the Fujitsu VP200 Series) length [62]. The memory-memory vector architecture (adopted by the CDC CYBER 205) differs from the previous one since a streaming vector unit is used instead of the vector registers, in order to avoid memory-register traffic. However, such a model has not been successful since the memory access is very time consuming, even if it is useful when dealing with large arrays. In fact, in these processors the special vector registers can store up to N array values at the same time and each operation performed on a single register is propagated to the whole set of values stored on it. Obviously, in order to speed up this mechanism, the availability of a fast link with the main memory is crucial.

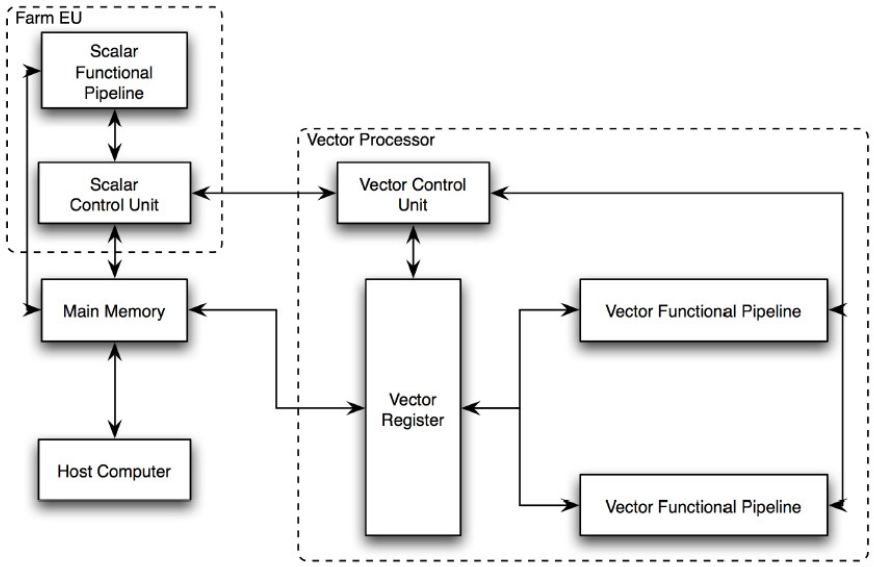

Figure 2.6: Vector architecture.

Array processors

An array processor performs only vector instructions, in fact the array processor architecture exploits the approach of executing concurrently the same instruction on several data. This approach guarantees high performances for programs involving a great number of vector instructions. An example of vectorial machine is represented by a vector co-processor (recently even PlayStation® and video cards are used) linked to an host-computer that loads in the main memory of the vector computer the program, together with the relevant data for the computation. The Instruction Unit (IU) fetches and decodes the instructions to be executed from the main memory and sends them to the Execution Units (EUs) (only if they involve arrays). The IU loads also into the Data Memories (DM) the operands involved in the array operations. Each EU-DM pair represents a Processing Element (PE). Such a model is sketched in Fig.2.7 and uses a distributed memory organization. As an alternative, we have the shared memory organization that is illustrated in Fig.2.8 which differs from the distributed memory organization because it can be simultaneously accessed by different EU with an intent to provide communication among them or avoid redundant copies.

Systolic array

A systolic array is a pipe network arrangement of processing units called cells. It is a specialized form of parallel computing, where cells (i.e. processors), compute data and store it independently of each other. A systolic array is composed of matrix-like rows of Data Drocessing Units (DPU) called cell which are similar to Central Processing Units (except for a program counter, since operation is transport-triggered, i.e., by the arrival of a data object). Each cell shares the information with its neighbours immediately after processing. The systolic array is often rectangular where data flows across the array between neighbour DPUs, often with different data flowing in different directions. The data streams entering and leaving the ports of the array are generated by Auto-Sequencing Memory units (ASM). ASM is an essential part of the anti machine paradigm. It is part of the instruction sequencer and is co-located with the datapath.

An example of a systolic algorithm might be designed for matrix multiplication. One matrix is fed in a row at a time from the top of the array and is passed down the array, the other matrix is fed in a column

ISSN: 2279–8773

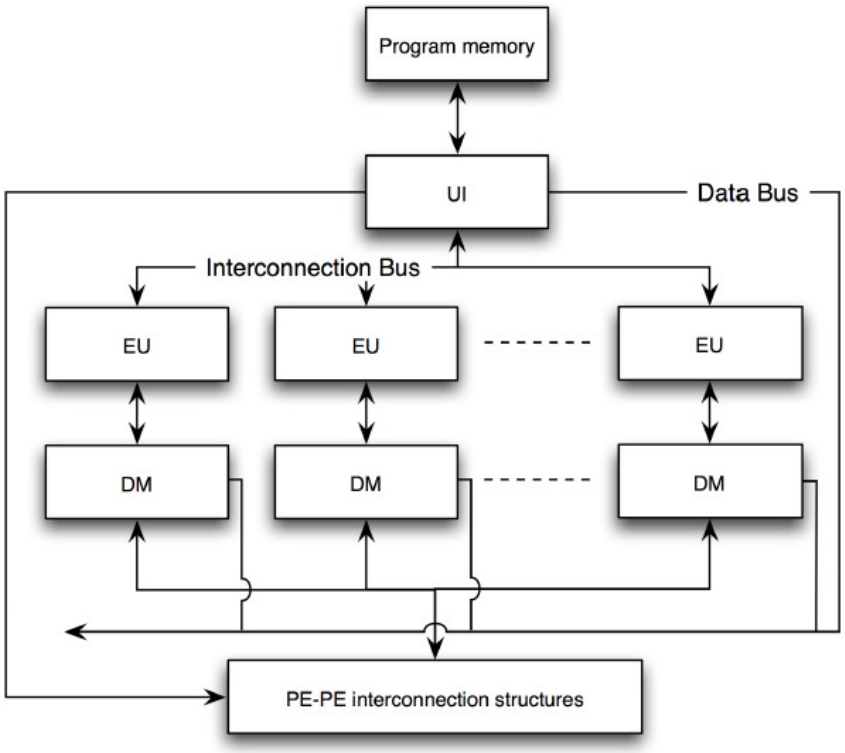

Figure 2.7: Array processor architecture with distributed memory.

at a time from the left hand side of the array and passes from left to right. Dummy values are then passed in until each processor has seen one whole row and one whole column. At this point, the result of the multiplication is stored in the array and can now be output a row or a column at a time, flowing down or across the array.

2.2.4 Limits of the sequential architectures

Scalar architectures described in the previous sections are hitting the limit of their performances and most of the modern scientific advances could never be met using sequential computers (as an example, the simulation of a climatic model for a period of ten years, needs the execution of 10^{16} floating point operations, that needs, if performed on a superscalar architecture, about three years). In fact, in addition to the execution speed, that can be increased using the just mentioned man-

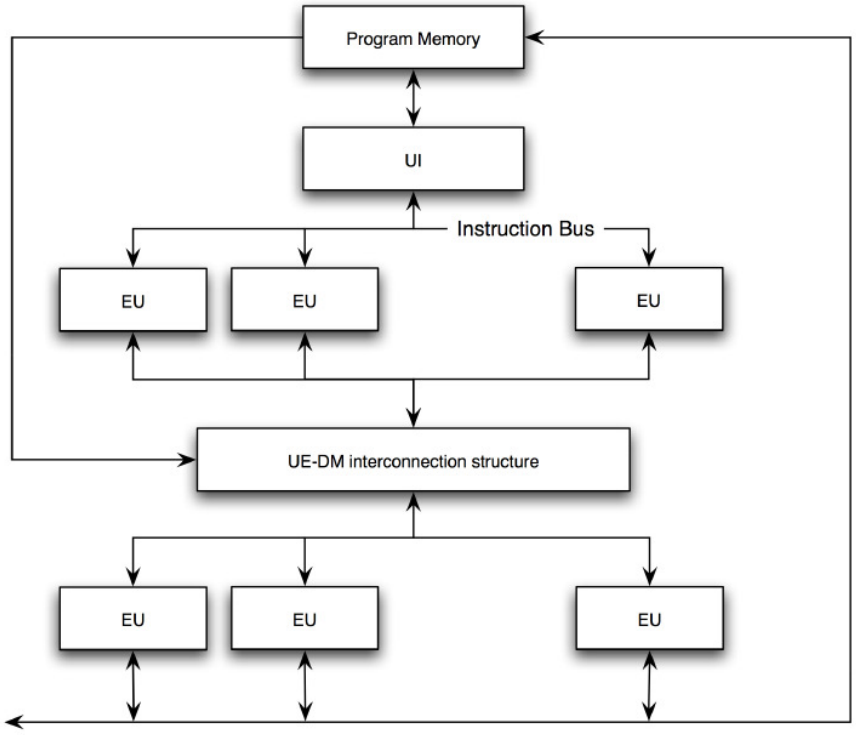

Figure 2.8: Array processor architecture with distributed memory.

agement concurrency that includes **ILP** and **DLP**, there is a physical limit in directly connecting the single CPU to a sufficiently large memory. Let us consider the case of a scalar computer that has to execute in one second the following cycle (Teraflops speed):

```
Do i= 1 to 1000000000000
     a(i)=b(i)+c(i)
EndDo
```

we need to transfer $3 \cdot 10^{12}$ variables from the memory to the CPU registers in one second. This implies that, if r is the mean distance of a word of memory from the CPU, the overall distance to be covered while transferring $3 \cdot 10^{12}$ variables in one second is $3 \cdot 10^{12} \cdot r$. Since the speed of light is $3 \cdot 10^8$ m/s one gets $r = 10^{-4}$ m. If we have $3 \cdot 10^{12}$ memory cells (each containing a word) packed as a matrix on a board around the CPU, then we have about 106 cells per row. This means that each cell cannot have a size larger than $10^{-6} \cdot r$ or 10^{-10}

m that is the mean dimension of an atom. Therefore, since we are not able to store a 32/64 bit number into a location of the size of an atom we cannot build a scalar computer with a peak performance of 1 T*flops*. This leads to the conclusion that, in order to increase hardware performances we need to build platforms having many processors each surrounded by a local memory.

2.3 Concurrency on multiple processor

As we have already mentioned parallelism can be achieved in several ways by exploiting different architectural features. To this end, it is very useful to introduce a classification (taxonomy) that labels various architectures by the nature of the adopted parallelism.

2.3.1 Flynn taxonomy

Starting from the von Neumann machine model, that consists of one processor which executes sequentially a set of instructions to produce a single result, a classification can be based on the concept of streams. Two are the basic streams of a computer: instructions stream and data stream. This represents the basis of the taxonomy proposed by Michael J. Flynn in the sixties that reads as follows:

> "*The multiplicity is taken as the maximum possible number of simultaneous operations (instructions) or operands (data) being in the same phase of execution at the most constrained component of the organization*" M.J. FLYNN, 1966 [63]

Following these considerations, the taxonomy introduced by Flynn is articulated as follows:

- **SISD** Single Instruction stream Single Data stream
- **SIMD** Single Instruction stream Multiple Data stream
- **MISD** Multiple Instruction stream Single Data stream
- **MIMD** Multiple Instruction stream Multiple Data stream

ISSN: 2279-8773

SISD

The SISD class of architectures (see Fig.2.9) is the simplest one. In fact, it consists of a sequential machine which computes only one instruction on a single data item. In other words, the SISD family consists only of von Neumann computers. This definition however, does not consider any parallel organization of CPUs, such as the integration of multiple elaboration units on the same chip that, as we have already seen, is a popular practical solution.

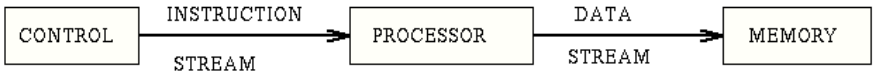

Figure 2.9: Sketch of SISD (Single Instruction, Single Data) machine.

SIMD

SIMD systems (see Fig.2.10) have a single control unit that executes one operation at time. They consist of several homogeneous processing units which can execute simultaneously that operation on different data sets. The central unit (control unit), in fact, acts like an emitter broadcasting the current instruction to be executed to the Processing Elements (PEs). **Cray 1**, **NEC SX-2**, **Fujitsu VP**xx and **APE** (or **QUADRIX**) are some examples of SIMD architectures. SIMD architectures can also be split in two classes:

- vector SIMD
- parallel SIMD

MISD

In the MISD case (see Fig.2.11) many instructions act simultaneously on the same data item. In this case, the granularity is represented by the processes (by the multiple instruction orograms). This architectural model has never been popular due to its complexity as well as its futility.

ISSN: 2279-8773

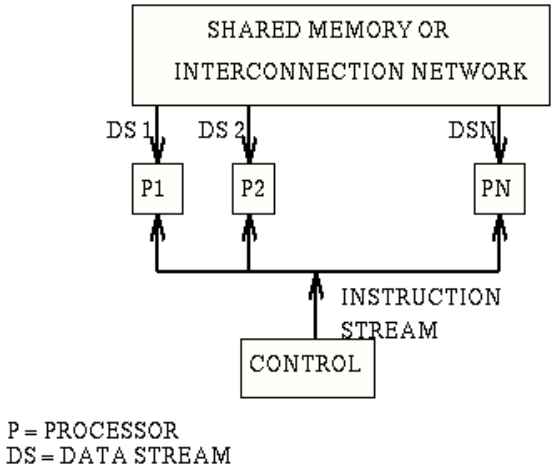

Figure 2.10: Sketch of SIMD (Single Instruction, Multiple Data) machine.

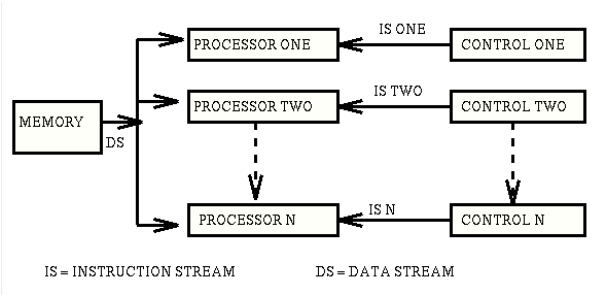

Figure 2.11: Sketch of MISD (Multiple Instruction, Single Data) machine.

MIMD

In MIMD machines (see Fig.2.12) the parallelism is exploited at a very coarse grain level on the execution tasks. This means that many PEs interpret different instructions on different data sets. Moreover, MIMD computers which make use of distributed memory are often referred to as thightly coupled machines (multiprocessors) while in case of a shared use of memory they are called loosely coupled machines (multicomputers). This class represents the multiprocessor version of SIMD architectures. Examples of computers belonging to MIMD multiprocessors class are ENCORE, MULTIMAX, SEQUENT & BALANCE; examples of computers belonging to MIMD multicomputers class are

ISSN: 2279-8773

INTEL iPSC and NCUBE/7.

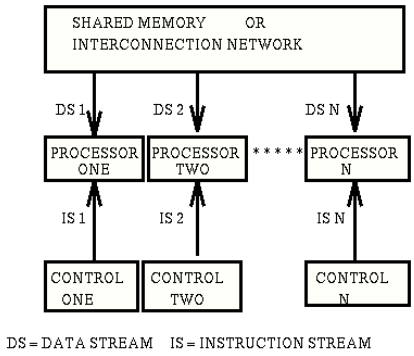

Figure 2.12: Sketch of MIMD (Multiple Instruction, Multiple Data) machine.

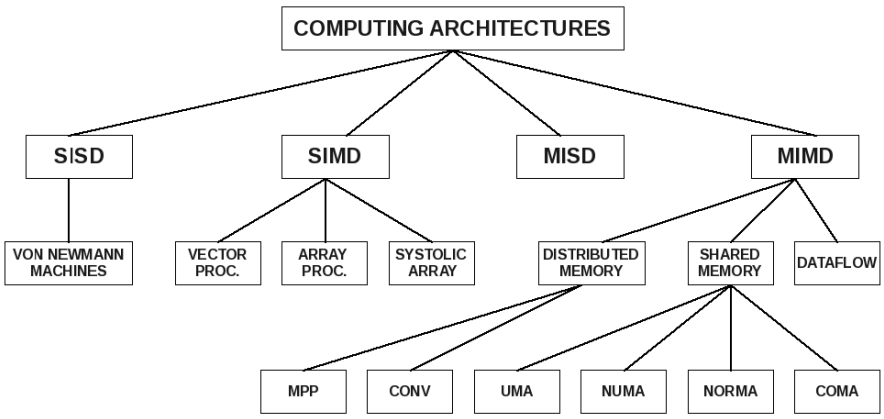

Figure 2.13: A summary sketch about computing architectures.

2.3.2 Other taxonomies

Since 2006, all the top 10 and most of the TOP500 supercomputers are based on a MIMD architecture.

Some further divide the MIMD category into the following categories:

Single Program, Multiple Data (SPMD) Multiple autonomous processors simultaneously executing the same program (but at independent points, rather than in the lockstep that SIMD imposes) on

ISSN: 2279–8773

different data. Also referred to as Single Process, Multiple Data, the use of this terminology for SPMD is erroneous and should be avoided, as SPMD is a parallel execution model and assumes multiple cooperating processes executing a program. SPMD is the most common style of parallel programming.

Multiple Program Multiple Data (MPMD) Multiple autonomous processors simultaneously operating at least 2 independent programs. Typically such systems pick one node to be the *host* (the explicit host/node programming model) or *manager* (the Manager/Worker strategy), which runs one program that farms out data to all the other nodes which all run a second program. Those other nodes then return their results directly to the manager. An example of this would be the Sony PlayStation 3® game console, with its SPE/PPE processor architecture.

Cell Cell microprocessor architecture (jointly developed by Sony, Sony Computer Entertainment, Toshiba, and IBM, an alliance known as STI) is a multi-core chip composed of one Power Processor Element (PPE) (sometimes called Processing Element, or PE), and multiple Synergistic Processing Elements (SPE) [64]. The PPE and SPEs are linked together by an internal high speed bus dubbed Element Interconnect Bus (EIB). Due to the nature of its applications, Cell is optimized towards single precision floating point computation. The SPEs are capable of performing double precision calculations, albeit with an order of magnitude performance penalty. Waiting for new chips to boost SPE double precision performance, a way to circumvent this is to use iterative refinement as software level, which means values are calculated in double precision only when necessary.

2.3.3 Memory architectures

As already mentioned above, Flynn's taxonomy is poor in classifying present parallel machines even if it is a very popular scheme. As already pointed out, the organization of the main memory is a key element in defining and classifying parallel architectures. By taking into account the memory structure of the machine (that is the organization of the main memory) three different models can be considered:

1. Distributed memory

ISSN: 2279–8773

2. Shared memory
3. Virtual Shared memory

This is particularly important for MIMD systems which exploit parallelism at coarse grain level. The organization of the memory can be, at an abstract level, either distributed or shared. The distributed memory model is illustrated in Fig. 2.14.

This model is referred in the literature as NORMA (*NO Remote Memory Access*). In this case the memory is distributed among the PEs, which are called computational (elaboration) nodes, each of which can be a multiprocessor. If all nodes are single processor machines, each node is made of a CPU that refers to its own memory. This means that a node can only access the memory directly attached to it. The communication between different nodes is managed by a message-passing interface, thanks to the interconnection network. In this case, however, the user has to implement the needed communication patterns, at the programming level. NORMA systems are generally made of many CPUs neither very powerful nor expensive.

An example of the distributed memory machine model of the NORMA type is the **IBM SP5** hosted at CINECA [65] in the early 2000 whose model with 64 nodes p5-575 interconnected with a pair of connections to the Federation HPS (High Performance Switch). Globally such machine had 512 IBM Power5 processors, capable of 4 double precision floating point operations per clock cycle, and 1.2 TBs of memory. The peak performance of SP5 is 3.89 TFlops. A p5-575 node contains 8 SMP processors Power5 at 1.9GHz. 60 nodes have 16GBs of memory each, 4 nodes have 64GBs each. The IBM-SP5 runs AIX 5.3 Operating System.

Quite different is the shared memory model. In this case the PEs coordinate their activity, accessing to data sets and instructions, in a global, shared memory environment. All processing elements have direct access to the whole memory space via the interconnection structure. Moreover, nodes can be dedicated or not. In this respect, it is useful to single out two categories: UMA (*Uniform Memory Access*) and NUMA (*Non Uniform Memory Access*). In the UMA scheme (see Fig. 2.15) the processors (**P** in the figure) work in an anonymous modality. Each process in a *ready* state (i.e. ready to be executed) is scheduled on the first available PE and the access time to the main memory (**M** in the figure) is the same for all PEs. On the contrary, in the NUMA scheme (see Fig. 2.16) the PEs work in a dedicated modality. Accordingly, each node gets an allocated partition of the global machine resources.

ISSN: 2279-8773

In particular, while the UMA model is closer to the shared memory scheme than the NUMA one, the NUMA model is made of subsets of processors with each subset having a local memory (**LM** in the figure) and I/O devices. The processors communicate among themselves via an interconnection structure which realizes a shared address space among the memory units. In other words, all the local memories share a global address space, accessible to the whole set of PEs (in other words, each PE can access, via remote operation, the local memory of the remaining PEs). Accordingly, local memory access time is shorter than in remote memory access. In the NUMA model the characteristics of both the simple shared memory and the distributed memory schemes coexist. This allows the NUMA model to go beyond the poor scalability of the UMA scheme when the number of processors and/or of the memory units (**M**) increases. In fact, when in a NUMA model the number of processors attempting to access the memory increases, the latency of the memory gets larger leading to progressively more severe bottlenecks. This problem is partially circumvented in the NUMA model by using the local memory of the processor (**LM**).

The last model of this family is the *virtual shared memory* (VSM) architecture. The VSM architecture has a memory organization similar to both the distributed and to the shared ones. In fact, each node refers to a MMU (*Memory Management Unit*) linked with both the local memory and an interconnection structure implementing the global address space. The MMU decides if a referred object has to be fetched from the local memory or from the remote one, once the processor address is known. The remote access is made possible by a support circuitry handling the communication, independently from the address of the processor. Sometimes these architectures are implemented as all cache machines. Such a model, in particular, is a special case of the NUMA in which local memories have been converted into caches. This model is also referred to as COMA (*Cache-Only Memory Architecture*).

2.3.4 The Interconnection infrastructure

The main difference between different MIMD architectures is concerned with the modality of data exchange. In fact, a MIMD system can be implemented in a variety of ways, depending on the adopted interconnection network, that represents a key point with respect to the performance in data exchange processes. Obviously, the choise of the

ISSN: 2279-8773

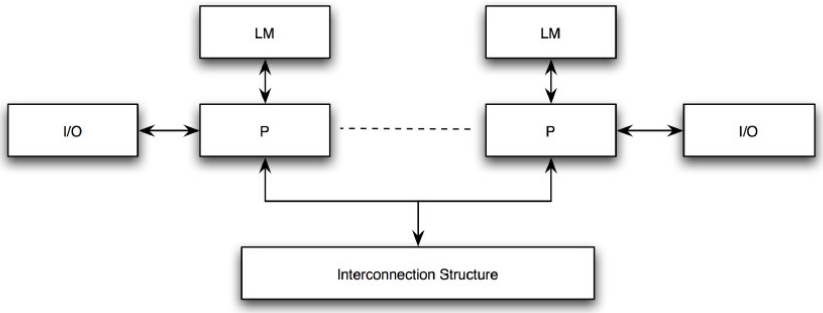

Figure 2.14: Distributed memory model.

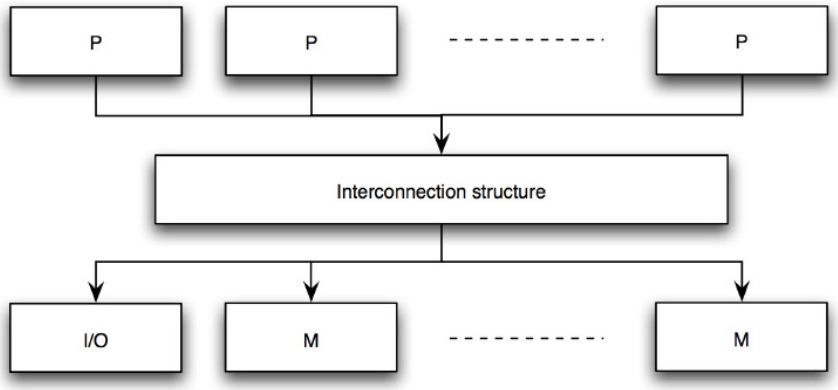

Figure 2.15: UMA model.

Message Passing paradigm to be used for data, information and signal exchange is also very important.

The main topologies of interconnection networks can be grouped as follows:

Full connection : this topology represents the most powerful interconnection system since each node is directly connected to the others. If we have N processors, each node has $N-1$ connections giving an overall total number of connections equal to $N(N-1)/2$. Obviously, despite the great advantage of having all the nodes simultaneously connected (the bandwidth is proportional to N^2), this model becomes impractical if N is large.

ISSN: 2279-8773

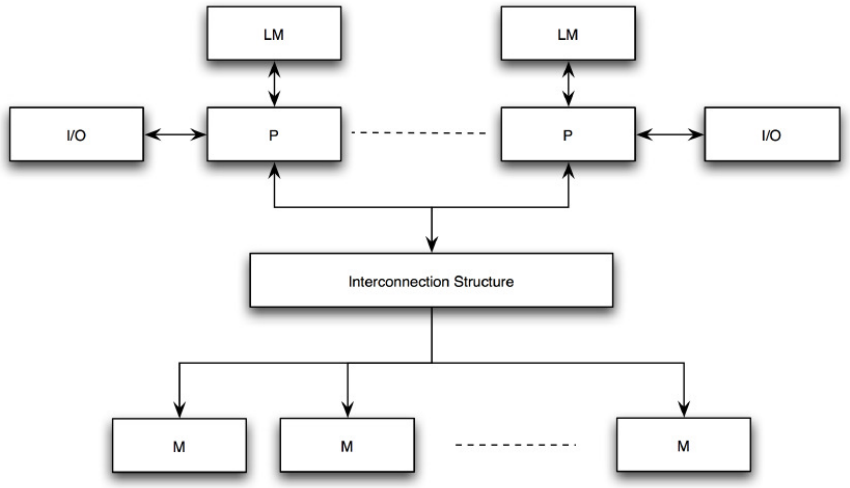

Figure 2.16: NUMA model.

Single shared bus : this type of network topology in which all of the nodes of the network are connected to a common transmission medium which has exactly two endpoints (this is the "bus", which is also commonly referred to as the backbone, or trunk). All data that is transmitted between nodes in the network is transmitted over this common transmission medium and is able to be received by all nodes in the network virtually simultaneously.

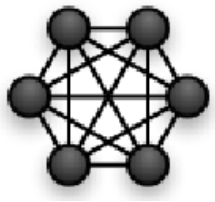

Figure 2.17: Full connection.

k-dimensional Grid (Mesh) : in such a network nodes are collocated on a grid of dimension k and width w, thus we have a total amount of nodes equal to w^k (see Fig. 2.19). Communications are performed only with neighbours (each node is interconnected to $2k$ nodes). Some versions of this topology present *wrap-around* connections between the border nodes (toroidal topology).

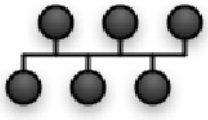

Figure 2.18: Bus.

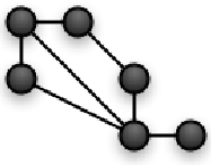

Figure 2.19: Mesh.

Tree and pyramid : a pyramidal network of dimension p is a complete quaternary tree with $\log_4 P$ levels, where the nodes of each level are connected by making use of a $2-D$ mesh (see Fig. 2.20).

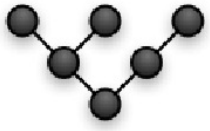

Figure 2.20: Tree.

Ring : it consists of a one dimensional array in which the final nodes are directly connected between them. In two dimension this is a thorus.

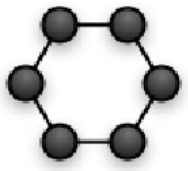

Figure 2.21: Ring.

Butterfly : a butterfly network has $(k+1)2^k$ nodes distributed among $(k+1)$ rows (ranks), each consisting of 2^k interconnected nodes.

ISSN: 2279–8773

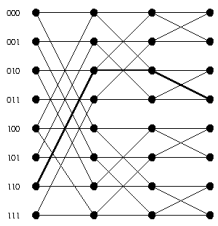

Figure 2.22: Butterfly.

Hypercube (Binary n-Cube) : such a topology consists of 2^k nodes arranged as an Hypercube of dimension k (see Fig. 2.23). The nodes are numbered from 0 to $2k-1$ and two nodes are connected only if their binary representations differ only for one bit.

Star : the type of network topology in which each of the nodes of the network is connected to a central node with a point-to-point link in a "hub" and "spoke" fashion, the central node being the "hub" and the nodes that are attached to the central node being the "spokes" (e.g., a collection of point-to-point links from the peripheral nodes that converge at a central node). All data that is transmitted between nodes in the network is transmitted to this central node, which is usually some type of device that then retransmits the data to some or all of the other nodes in the network, although the central node may also be a simple common connection point (such as a "punch-down" block) without any active device to repeat the signals.

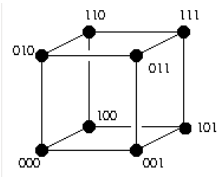

Figure 2.23: Hypercube.

Cross-bar switch : a crossbar switch is a switch topology where every node can be connected to any other node in the system. This topology has traditionally been used for high-performance computing systems since it can provide concurrent independent data paths between pairs of nodes in the system for maximum total system bandwidth. This topology is also flexible because the connec-

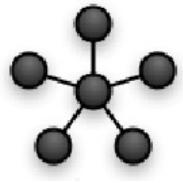

Figure 2.24: Star.

tions between node pairs can be changed dynamically as needed for optimum system communication.

Network of computers : this is the case of a single communication channel shared by all nodes of the system. Each node can be either a workstation or a PC (or both if we are not interested into having homogeneity) and the interconnection network is, typically, a LAN. Although as an individual platform this model is rather poorly performing, it is the reference scheme for Grid computing. In the case of Grid computing each computing apparatus is considered as a working unit and the public network must provide access for work coordination.

Clusters : this is a particular cost effective platform representing the most popular trend in concurrent computing. Clusters are essentially farms i.e., a set of independent servers each connected to a central unit, called front end, that is directly interfaced with the outer network. The nodes are normally PC like units having good memory sizes and equipped with low cost hard disks. The interconnection network consists of a switch, typically of Gigabit technology. This solution makes it feasible to build a parallel system in a simple and cheap way.

2.4 Concurrent computing

Along with the evaluation of computer architectures and platforms, program design and organization models are needed to provide users with suitable tools to implement concurrency in their applications.

The basic sequential machine programming paradigm is the execution of a set of instructions. An instruction can specify, in addition to various arithmetic operations, the address of data to be read or written in memory and/or the address of the next instruction to be executed.

While it is possible, in principle, to program a computer using this basic scheme directly in machine language, this is for most purposes impractical since one needs to keep track of millions of memory locations and manage the execution of thousands of machine instructions. Hence, modular design techniques are applied, whereby complex programs are constructed from simple components, and components are structured in terms of higher level abstractions such as data structures, iterative loops, alternative sequences and procedures. Abstractions (like the procedures) make the exploitation of modularity easier by allowing objects to be manipulated without concern for their internal structure. So do high-level languages such as Fortran, Pascal, C, C++. These high level (artificial) languages allow program design expressed in terms of abstractions to be translated automatically into executable code.

On the other hand, parallel programming introduces additional sources of complexity with respect to sequential programming if we were to program at the lowest level. In this case, in fact, not only would the number of instructions to be executed increase, but the execution of thousands of processes and the coordination of millions of interprocess interactions would also need to be managed explicitly. Hence, abstraction and modularity are at least as important as in sequential programming. In order to develop a parallel application one has to look at well established programming paradigms. Three are the main paradigms in parallel computing:

Message passing: this is probably the most widely used parallel programming paradigm today. Message-passing applications create multiple tasks, with each task encapsulating local data. Each task is identified by a unique name, and tasks interact by sending and receiving messages to and from named tasks. The message-passing paradigm does not preclude the dynamic creation of tasks, the execution of multiple tasks per processor, or the execution of different programs by different tasks. However, in practice most message-passing systems create a fixed number of identical tasks at program startup and do not allow tasks to be created or destroyed during program execution.

Data parallelism: another commonly used parallel programming paradigm that calls for the exploitation of the concurrency deriving from the application of the same operation to multiple elements of a data structure. A data-parallel application consists of a sequence of such operations. As each operation on each data element

ISSN: 2279-8773

can be thought of as an independent task, the natural granularity of a data-parallel computation is coarse, and the concept of locality does not arise naturally. Hence, data-parallel compilers often require the programmer to provide information about how data are to be distributed over processors, in other words, how data are to be partitioned into tasks. The compiler can then translate the data-parallel program into an SPMD formulation, thereby generating communication code automatically. The implementation of the data-parallel paradigm is represented, as an example, by the High Performance Fortran (HPF) parallel programming language.

Shared memory: for this programming paradigm tasks share a common address space, which they read and write asynchronously. Various mechanisms such as locks and semaphores may be used to control access to the shared memory. An advantage of this paradigm from the programmer's point of view is that there is no notion of data ownership, and hence there is no need to specify explicitly the communication of data from producers to consumers. This paradigm can simplify program development. However, understanding and managing locality becomes more difficult and this is an important consideration that need to be considered on most shared-memory architectures. It can also be more difficult to write deterministic programs.

In this section only the message passing paradigm will be taken into account since it represents the *de-facto* standard in directive parallel programming.

2.4.1 The *a priori* design of a parallel application

Regardless the adopted paradigm, the methodological design of a scalable parallel application must follow well determined steps. In fact, most programming problems can be tackled using different parallel approaches. The best solution may differ from that suggested by existing sequential algorithms. The design methodology described here and proposed by Ian Foster [66] is intended to foster an exploratory approach in which machine-independent issues (such as concurrency) are considered early and machine-specific aspects are delayed until late (or if possible left with specific software of the machine). This methodology structures the design process as four distinct stages: partitioning, communication, agglomeration, and mapping, as sketched in Fig. 2.25.

ISSN: 2279–8773

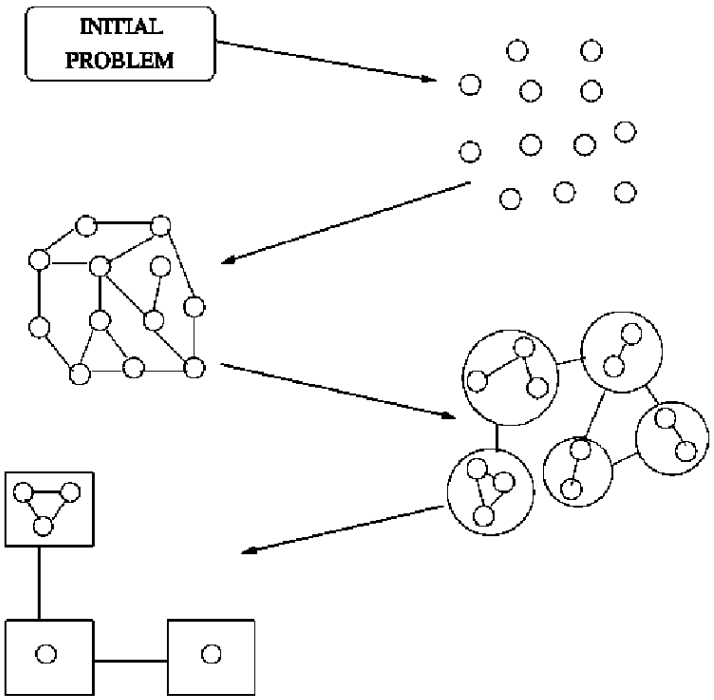

Figure 2.25: A design methodology for parallel applications.

Partitioning

The computation that is to be performed and the data involved are decomposed into small tasks. Practical issues such as the number of processors in the target computer are ignored, and attention is focused on recognizing opportunities for parallel execution. In particular, this stage consists in defining the computational grain of the parallel application, taking care of pushing the partitioning at the lowest level ensuring a good flexibility. Such a partition can be undertaken at both data and computation levels:

- *domain decomposition:* this technique first decomposes the data associated with a problem and then partitions the computation that is to be performed, typically by associating each operation with the data on which it operates.
- *functional decomposition:* this technique represents a different and complementary way of thinking about problems. In this approach, the initial focus is on the computation that is to be performed

ISSN: 2279-8773

rather than on the data manipulated by the computation. After being successful in dividing the computation into disjoint tasks, one proceeds into examining the data requirements of these tasks. These data requirements may be disjoint, in which case the partition is complete. Alternatively, they may overlap significantly, in which case considerable communications will be required to avoid replication of data.

Communication

Once the partition into very small tasks of data and computations has been performed, the communication required to coordinate the execution of the various tasks needs to be determined, and appropriate communication structures and algorithms need to be defined. The tasks generated by the partitioning can, in general, execute concurrently though they cannot execute independently. Usually the computation to be performed in one task will require the communication of data associated with another task. Communication requirements differ depending on whether one has performed a domain or a functional decomposition. In fact in the former case, the implementation of communications between different tasks can be difficult establish since partitioned data often remains tightly coupled. On the contrary, if a functional decomposition has been adopted, communications can be easily decoupled.

Agglomeration

The tasks and communication structures defined above need to be evaluated with respect to performance requirements and implementation costs. This may request that individual tasks are combined into larger tasks to improve the performance or to reduce the development costs. At this stage one has to decide if the granularity, derived by the partitioning stage (in which as many tasks as possible have been defined), is acceptable or if one has to push the granularity to a coarser level. In general, one looks for tasks (and communications among them) to be associated into greater tasks. At the end of this stage it is advised to minimize the overall communication and to obtain a number of tasks larger than the number of processors.

ISSN: 2279-8773

Mapping

Tasks are assigned to the various processors in a way that attempts to maximize the processor utilization and minimize the communication costs. The main goal of this stage is to ensure an optimal load-balancing between different nodes (processors) of the platform. In this final stage we have to decide where each task has to be executed, depending on the used platform. In general two strategies can be adopted:

- Tasks which can be executed independently are mapped on different physical processors
- Tasks which keep a high degree of coupling are mapped on the same processor.

Clearly, these two strategies might sometimes conflict. In this case the design will require some tradeoffs. Also, resource limitations may restrict the number of tasks that can be mapped on a single processor or the number of processors that the tasks may use. In order to perform the mapping of the tasks there are many load balancing algorithms that help the user. Obviously the user based on his/her knowledge of the application may decide to apply one of them like recursive bisection, local algorithm, probabilistic method, etc. The user can also choose to leave to the Load Balance utility of the machine in order to take care of the problem. Load Balancing struggle to avoid the unshared state in processors which remain idle while tasks compete for service at some other processor equalizing the load on all processors. Algorithms for load balancing have to rely on the assumption that the on hand information at each node is accurate to prevent processes from being continuously circulated about the system without any progress. This is one of prerequisites to utilize the full resources of parallel and distributed systems. Load balancing may be centralized in a single processor or distributed among all the processing elements that participate in the load balancing process.

Such load balancing algorithms are the *Recursive Bisection*, the *Local algorithm*, the *Probabilistic Method*, the *Round-Robin* and the *biasing algorithm*.

2.4.2 Models of parallelism

The *a priori* design of a parallel application is driven by the adoption of some standard models of parallelism. These models can be classified

ISSN: 2279-8773

on the basis of the exploited parallelism:

Flux parallelism (predetermined interactions between data):

- *seq* in which parallelism is not exploited
- *farm* in which each process is assigned a determined work-load, by a central unit that manages the distribution of tasks. This model is also called Master-Slave or Master-Worker paradigm)
- *pipe* the whole computation is partitioned into different stages each of which is assigned to a process as in a chain. Each stage of the pipe can be parallelized at a finer grain
- *loop* in which a block of processes is iterated in time
- *data parallelism* (few dependencies among data and there is no need for a great number of interactions):

- *map* completely independent data
- *reduce* subsequent associative reduction of data
- *comp* sequential composition of modules with some interactions

In Fig. 2.26 a sketch of the *farm* model is given.

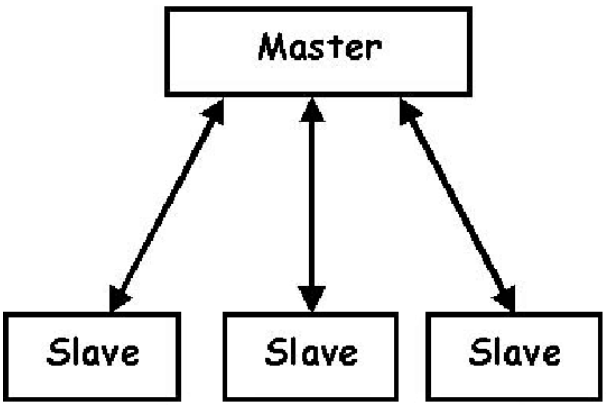

Figure 2.26: The farm model.

2.4.3 Tools for parallel programming: MPI

As already pointed out above, an important paradigm for parallel computing is Message Passing. This paradigm has been developed in order

to be used on distributed memory platforms although it can be also implemented efficiently on a shared memory architecture. In the message passing paradigm a computation consists of one or more processes that communicate by calling library routines to send and receive messages to/from other processes. In this respect, two are the main libraries that implement the paradigm: PVM (Parallel Virtual Machine) [67–69] and MPI (Message Passing Interface) [70]. We deal in detail here with MPI (in particular, MPI-1 implementation) both because MPI is increasingly becoming the standard library both in homogeneous (MIMD and MIMD-like platforms) and heterogeneous (GRID systems) environments and because of the work made on it by our research group for the European project EGI-Inspire [71]. Accordingly, we can summarize the main goals of MPI as follows:

- provide source-code portability
- allow efficient implementations
- offer a great deal of functionality
- support heterogeneous parallel architectures

In the message passing programming model, each process has a local memory and no other process can directly read from or write to that local memory. Parallel programming by definition involves cooperation between processes to solve a common task. There are two sides to the question of programming processes that cooperate each other. The programmer has first to define the processes that will be executed by the processors and then to specify how those processes synchronize and exchange data with one another (as already specified at an abstract level in the previous section). A central point of the message-passing model is that, obviously, the processes communicate and synchronize by exchanging messages. As far as the processes are concerned, the message passing operations are just calls to a message passing interface that is responsible for dealing with the physical communication network linking the processors.

Prior to exchanging messages it is needed to determine the "communication universe", in which point-to-point or collective operations can act, by defining a communicator. Each communicator contains a group of processors and the source and the destination of a message is identified by the process rank within that group. In MPI there are two types of communicator here discussed

ISSN: 2279–8773

- **Intracommunicator** used for communicating within a single group of processes,
- **Intercommunicator** used for communicating within two or more groups of processes (in MPI-1, an intercommunicator is used for point-to-point communication between two disjoint groups of processes).

Messages are central to the message passing programming model. They are exchanged between processes. When two processes exchange a message, data is copied from the memory buffer (or memory locations) of one process into the memory buffer of the other process. The data sent in a message comes under two headings: *contents* and *envelope*. The contents of the message are pure user data and are not interpreted neither by the communication interface nor by the communication system that lies behind that interface. The data on the envelope, however, is used by the communication system to manage the copying of the content of the message between local memories.

The simplest form of message is a *point-to-point* communication in which a message is sent from a sending to a receiving process. Only these two processes need to know anything about the message. The communication itself consists of two operations: send and receive. The send operation can be either synchronous or asynchronous depending on whether or not it is completed[1] before or after the corresponding receive operation has started. *MPI point to point* prinitives can be classified in blocking or non blocking procedures. The blocking ones return the control only when the corresponding communication is completed. The non blocking ones return the control straight-away and allow the process to continue performing other work. Many message-passing systems do also provide primitives allowing large numbers of processes to communicate. Such primitives implement the so-called *collective communications* and they are of blocking type. Examples of these are barrier, which synchronizes the processors, broadcast, which allows a one to many communication and reduction operations which takes data items from several processors and reduces them to a single data item that may or may not be made available to all of the participating processes. Moreover, the library includes primitives which are particularly designed to perform a collective domain decomposition.

[1] the completion of the communication means that memory locations used for the message transfer can be safely accessed. MPI communication modalities differ from the conditions needed to completion.

ISSN: 2279–8773

For example, the most commonly used are:

- scatter (*one-to-all communication*): different data are sent from the root process to all the others in the MPI communicator (including the root process)
- gather (*one-to-all communication*): different data are collected by the root process from all other processes in the communicator (including the root process). It is the opposite of the scatter primitive.

A synopsis of the most commonly used MPI primitives are listed in Table 2.1.

Table 2.1: Most common MPI primitives.

Primitive	Description
MPI_comm size	Creates the communicator processes
MPI_comm rank	Gives the rank to each process of the communicator
MPI_send	Sends a message to another process
MPI_recv	Receives a message from another process
MPI_barrier	Blocks the caller until all processes have called it
MPI_bcast	Broadcasts a message form one process to all processes
MPI_gather	Each process sends a message to a root process that receives and stores it in rank order
MPI_scatter	The inverse operation of gather
MPI_reduce	Combines the elements of an incoming data using a pre-determined operators in the root memory space

Clearly, despite its great portability and diffusion, MPI shows some severe limitations:

1. the management of the communication is entirely on the hands of the programmer
2. the library does not provide models and tools for efficiency evaluation (e.g. to predict the scalability of the application on the basis of the implemented granularity) and a profiling of the program in this sense is needed;

ISSN: 2279-8773

3. portability is sometimes difficult and a deep restructuring may be necessary to implement the parallel code;
4. due to its explicit nature, MPI is *error-prone*.

A pictorial representations of Broadcast, Gather and Scatter primitives are given in Fig. 2.27.

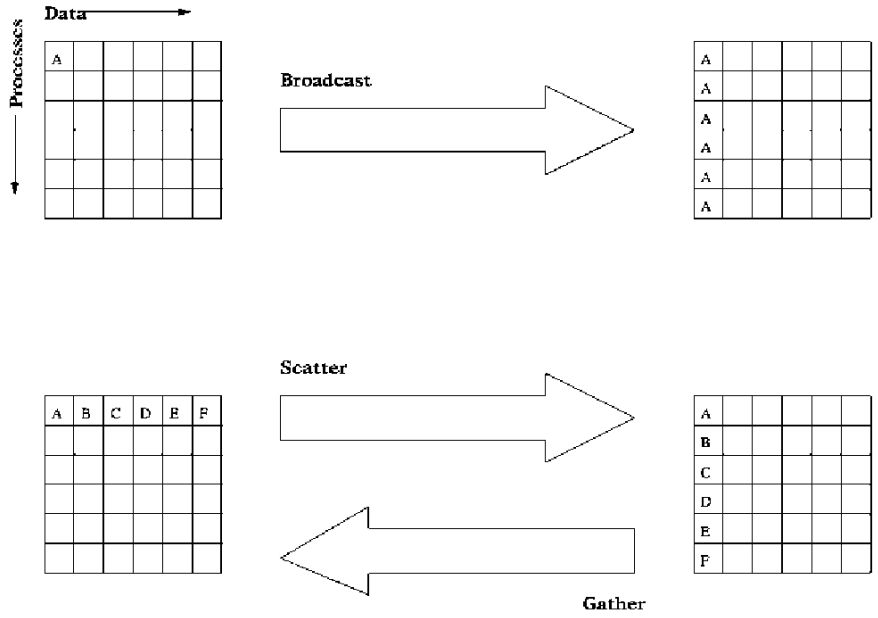

Figure 2.27: A pictorial view of some collective primitives.

Charm++

It has been developed at the University of Illinois, a machine independent parallel programming language written in C++ called Charm++. The design of the system is based on the following tenets:

- Efficient Portability: Charm++ programs run unchanged on MIMD machines with or without a shared memory. The programming model induces better data locality, allowing it to support machine independence without losing efficiency.
- Latency Tolerance: Message-driven execution, supported in Charm++ is a mechanism for tolerating or hiding communication latency. In message driven execution a processor is allocated to a process only

ISSN: 2279–8773

when a message for the process is received. This means that when a process blocks, waiting for a message, another process may execute on the processor.

- Dynamic Load Balancing: Charm++ provides dynamic (as well as static) load balancing strategies.

The package consists of potentially medium-grained processes (called chares), a special type of replicated process, and collections of chares. These processes interact with each other via messages. There may be thousands of medium-grained processes on each processor, or just a few, depending on the application. The "replicated processes" can also be used for implementing novel information sharing abstractions, distributed data structures, and intermodule interfaces. The programming language can be considered a concurrent object-oriented system with a clear separation between sequential and parallel objects.

2.4.4 The evaluation of performances and scalability

Execution time is not always the most convenient metric to use to evaluate parallel algorithms performance. Given that the first goal of parallel programming is make an application faster than its sequential implementation, one needs some parameters able to quantify the achieved performance. The most popular parallel performance parameters are *Speed up* (S) and *Efficiency* (E). The speed up S is calculated as:

$$S(n) = \frac{T_s}{T_p} \tag{2.1}$$

where T_s represents the execution time of the best sequential implementation of the code and T_p is the execution time of the related parallel implementation on n processors. The speed up is a pure number and its maximum value is equal to the number of processors used in the calculation. The corresponding efficiency (E) is given by:

$$E = \frac{S(n)}{n} \tag{2.2}$$

The efficiency can be seen as the fraction of time that the processors spend in doing useful work. Sometimes it may happen that the calculated speedup is greater than the number of processor. This behaviour

ISSN: 2279-8773

is called *superlinear*. This effect is usually related to a better use of fast memory, such as cache memory. In fact, if the parallelization of the application allows the data to be kept in cache (rather than in memory) during the calculation, costs of memory access decreases to make the efficiency greater than 1 (and the speedup superlinear).

A model able to predict the theoretical maximum speed-up using multiple processors is the Amdahl's law which compares the expected speedup of the parallelized implementation of an algorithm with that of the serial algorithm, under the assumption that the problem size remains the same when parallelized. More in detail, if P is the fraction of a program that can be parallelized and $(1 - P)$ is the fraction that cannot be parallelized (i.e. the strictly serial fraction), then the maximum speedup that can be achieved using n processors is given by the equation

$$S(n) = \frac{1}{(1 - P) + \frac{P}{n}} \tag{2.3}$$

2.5 Concurrency on the network: the Grid

In very recent times Grid computing has emerged as a new important field of concurrent computing in heterogeneous environments. Grid computing is focused on large-scale resource sharing, innovative applications and, clearly, high throughput computing. The Grid concept was effectively born in the mid '90s and it has been defined as [72]:

> *A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities.*

In this respect, the Grid deserves to be considered separately from parallel computing. Grid computing is not aimed at high-performance computing. It is rather focused on high-throughput computing for complex computational applications which need the gathering of a large ensemble of computer resources and expertises. Accordingly, the basic problem lying behind the Grid concept is related to the coordinated sharing of the resources. This means easy access to all kind of available computing platforms, software, data and other resources like human skills.

ISSN: 2279–8773

2.5.1 The foundations of Computer Grids

Computer Grids are characterized by the heterogeneous nature of their wide ensemble of computing hardware components and by the composite nature of the applications considered. Therefore the establishing of a production Grid requires the implementation of several components.

The first characteristic feature of a Grid is the composition of different expertises related not only to the involved hardware, but also to the specific components of the problems tackled. The management of these competence based instrumentation and tools requires specific attention when building demonstrators for the Grid.

The second critical feature is given by the nodes of the Grid. The user can make direct use of just one type of computer, typically a desktop workstation, not only to write, debug and compile the codes but also to launch the simulation. A bunch of high performance computers or supercomputers can be chosen to take care of running the codes and crunching the data fed to them. Other computers may be used either to take care of rendering the results in a graphical form or to power a virtual reality display device.

The third feature of a Grid is the communication software to make the whole collection of codes user-friendly. Communication software bridges all of the gaps, between different computers, between computers and people, even between different people. This turns the physical connections between computers from that of a collection of individual machines into that of an interconnected computing system.

The fourth feature of the computational Grid is the physical network that links the various machines. Networks with high bandwidth and low latency are the most favored to provide rapid and reliable connections between the machines. To actually communicate over these physical connections it is also necessary to have some smart communication software running.

Given that we have an interconnected, communicating network of computers, processors with memory, one further component is needed. This fifth feature is something like an operating system that can be used to configure, manage, and maintain the Grid computing environment. This virtual environment needs to span the extent of the computational Grid and makes it usable by both administrators and individual users. Such an environment will enable machines and/or instruments that may be located in the same building, or separated by thousands

ISSN: 2279-8773

of miles, to appear as one system. This virtual environment, therefore, must provide the administrators all the functions which allow her/him to time the system management tools to deal with a changing heterogeneous platform. This software needs also, to speak, on the side of the user, the language he is used to and has, therefore, to fall into the category of problem solving environments (PSE). More in detail such an environment has to allow the user to get the best from the platform both in preparing input data and at running time with no need to get involved in managing related technicalities. The most popular denomination of this type of software is "middleware" and still needs a significant amount of designing and implementing.

The main middleware distributions currently used in European production grids are ARC [73], gLite [74] and UNICORE [75]. They support now a large number of user communities with complementary requirements and dimensions, ranging from teams of a few individuals to very large international collaborations with thousands of researchers and tens to hundreds thousand jobs daily. These middleware solutions have become a reference in many countries also outside Europe (Asia-Pacific, Africa, India, China, South America, etc.) thanks to the efforts of EC-funded international projects. Presently, the european standard in Grid computing is represented by the gLite middleware [74], (that derives from the Globus middleware [76] and has been developed at the Argonne National Laboratory) developed through collaborative efforts of more than 80 people in 12 different academic and industrial research centers as part of the EGEE Project [77] (see later for more detail).

Although the existing middleware has demonstrated the ability to support a production infrastructure, there is also clear evidence that it will need to evolve in response to the evolution of technology and to the continuous flow of user and operation requirements in areas such as functionality, robustness, usability, deployment, adherence to standards, interoperability with other infrastructures, with the additional constraint to maintain interface stability. For the above mentioned reasons, the European Middleware Initiative (EMI) [78] project has been established to deliver a consolidated set of middleware products based on the four major middleware providers in Europe - ARC, dCache [79], gLite and UNICORE. The products, managed in the past by these separate providers, and now developed, built and tested in collaboration, are for deployment in EGI [4](as part of the Unified Middleware Distribution or UMD [80]), and other distributed computing infrastructures, extend the interoperability and integration between grids and

ISSN: 2279-8773

other computing infrastructures, strengthen the reliability and manageability of the services and establish a sustainable model to support, harmonize and evolve the middleware, ensuring it responds effectively to the requirements of the scientific communities relying on it.

2.5.2 The EMI Project

The mission of EMI is to:

- deliver a consolidated set of middleware components for deployment in EGI [4] (as part of the Unified Middleware Distribution - UMD), PRACE and other DCIs,
- extend the interoperability and integration with emerging computing models,
- strengthen the reliability and manageability of the services and establish a sustainable model to support,
- harmonise and evolve the middleware, ensuring it responds effectively to the requirements of the scientific communities relying on it.

For several years there have been many issues surrounding the integration of Open Grid Services Architecture [81] (OGSA) concepts in Distributed Computing Infrastructures (DCIs) and standardisation of Grid services within Grid middleware systems provided by EMI. OGSA represents a massive architecture in the context of distributed systems based on the concept of Grid services that cover functionalities of many technical areas such as compute, data, and security. The issues can be partly explained by the fact that working interactions among numerous Grid services as envisaged by OGSA are non-trivial when we consider their implementations based on Web services message exchanges. These exchanges essentially represent an XML-based [82] Remote Procedure Call (RPC) where each single difference in the correspondingly used XML-based protocol or schema can break the interconnection between EMI services and their clients. Standardisation of these XML-based protocols or schemas bears the potentiality of enabling more functioning and stable interconnections between the EMI Grid services which form together a DCI (thus enabling interoperability) with other DCIs and similar infrastructures which adopt the same standards.

ISSN: 2279-8773

Such aforementioned common open standards can be defined as follows: A common open standard is any standard developed by a standardization development organisation (SDO) following an open process and being commonly relevant to the e-Science community. Those standards are typically normatively defined and publicly available. In principle, EMI works with normatively defined in specifications standards while other standards are still considered as emerging open oned because they are not implemented by many technology providers yet.

The EMI middleware is addressing four technical areas: Computing, Data, Security and Infrastructure.

Compute Area: include middleware services and corresponding client components involved in the processing and management of user requests concerning the execution of a computational task. They cover the interaction with Local Resource Management Services, the provision of a common interface to the computational resources of a site (the so-called Computing Element) and the availability of high-level meta-scheduling, workflow execution and task tracking functionality.

Data Area: include middleware services and corresponding client components involved in the processing and management of user requests concerning storage management, data access and data replication. Multiple storage solutions exist, addressing different types of resources (disk, tape or a combination of the two) all exporting the same Storage Resource Management (SRM) interface (the so-called Storage Element). Data access libraries are available for storage systems not offering a POSIX interface towards the computational resources. Data and metadata catalogues track the location of copies of data chunks in multiple places.

Security Area: include middleware services and components that enable and enforce the Grid security model, allowing the safe sharing of resources on a large scale. They cover identity management, Virtual Organization membership management, authentication, delegation and renewal of credentials, and authorization.

Infrastructure Area: include middleware services and components that provide common information and management functionality to deployed Grid services. They include the Information System and Service Registry, which provide a view of the services available on the Grid together with their characteristics; the messaging infrastructure, that allows to collect and distribute messages generated

ISSN: 2279–8773

by Grid services or user tasks; the service monitoring and management providers that allow the retrieval of Grid services status information and service state management; the Logging and Bookkeeping services that allows collecting, aggregating and archiving job execution information; the accounting functionality to collect, distribute and publish information concerning the usage of resources This area also deals with internal infrastructure components, such as service containers t hat are required for middleware services.

The EMI middleware adopts many software components that implement the above described services.

2.5.3 Towards the European Production Grid

Under the pressure of the Physics community that, at that time, was planning the Large Hadron Collider (LHC) experiment at CERN, the European Union funded within the Framework Programs 6 and 7 (FP6 and FP7) the design and the construction of a Europe wide production Computing Grid. The first project to be approved was the "Research and Technological Development for an international Data Grid" in the year 2001 with a budget of 12 million €. Then on April 2004 the already mentioned project called EGEE (European Grid for E-science in Europe) [77] has been first approved for two years under the coordination of the technology division of CERN. This 24 months project of FP6 had a cost of over 46 million € and was aimed at gathering a high capacity complex distributed system to support the LHC experiment. It was based on a consortium of over 70 institutions of 27 Countries. Its success went well over the goals of the LHC experiment and the boards of the European Union. Because of this, the project was renewed twice for additional two years each time (EGEE II and EGEE III respectively) and the acronym dropped its "in Europe" specification to reflect the new global mission of the project. As a matter of fact, the importance of e-Science for researchers in academia and industry became more apparent and the benefit to innovation from the EGEE e-Infrastructure that simultaneously supports applications from various scientific areas, providing a shared pool of resources, independent of geographic location, with round-the-clock access to major storage, compute and networking facilities became more substantial. The EGEE project significantly extended and consolidated its infrastructure to link national,

ISSN: 2279-8773

regional and thematic Grid resources and interoperate as well as with other Grids around the globe. The resulting high capacity, world-wide infrastructure rapidly surpassed the capabilities of local clusters and individual centres, providing a unique tool for collaborative computing in science ("e-Science"). So far, several large- and small-scale communities use the EGEE infrastructure as an every-day tool for their work. Applications deployed come from High Energy Physics, Life Sciences, Earth Sciences (including the industrial application EGEODE), Astrophysics, and Computational Chemistry. EGEE has expanded the portfolio of supported applications to include Nuclear Fusion as well as other disciplines.

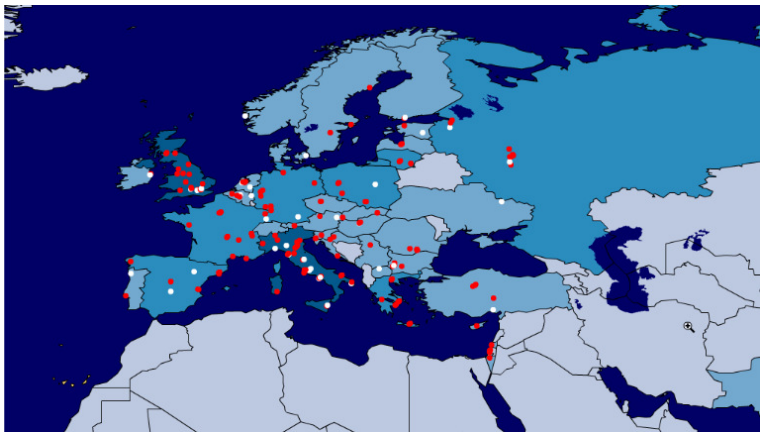

Figure 2.28: Partner countries of the EGEE project.

In the end EGEE has established a Consortium consisting of more than 90 partners from 32 countries, grouped into 13 federations and representing almost all major and national Grid efforts in Europe (see Fig. 2.28) as well as projects from the US and Asia. In addition, a number of related projects have extended the infrastructure further, to the Mediterranean area, Baltic States, India, Latin America and China. Combined with other related projects spurred out from or affiliated with EGEE, this project has played around the world. The project has managed also to constitute organized Virtual communities (called Virtual Organizations or VO) devoted to gather together in a structured way the members of a given scientific community.

Given the success of the EGEE programme, it has become unavoidable to build on its achievements and prepare the transition towards a sustainable infrastructure which the main actor is, nowadays, the al-

ready mentioned European Grid Infrastructure (EGI) [4]. The design of EGI and of the procedure for implementing it has been the goal of the European Grid Initiative Design Project (EGI_DS) [83], approved under FP7, aimed at implementing the required structural changes that allowed a seamless transition to EGI that established its headquarter in Amsterdam (March 2009) and has now become EGI.eu (february 2012). On that date the original word "Initiative" was changed into "Infrastructure" to visibly indicating the fact that EGI is a solid reality ensuring the continued provision of the Grid service. EGI started officially in May 2010 and it is a European coordination body managing the relationships with the National Grid Infrastructures (NGI). In Italy the NGI, named IGI (Italian Grid Initiative) [84], is at present a Joint Research Unit (JRU) still waiting for formal recognition.

The main stakeolders of EGI are the NGIs, which provide on national level the services for a seamless, shared and uniform access to a variety of computing resources, ranging from PC clusters to sites also operating supercomputers and all sorts of scientific archives (see Fig.2.29 for details). The goal of EGI is to link existing NGIs together and to actively support the set-up and initiation of new NGIs in those countries, where corresponding efforts do not yet exist. The characteristics of the NGIs can be identified as follows:

1. be the only recognized national body in a country with a single point-of-contact representing all institutions and research communities related to a national grid infrastructure;
2. have the capacity to sign the statutes of EGI.org, either directly or through a legal entity representing it;
3. have a sustainable structure, or be represented by a sustainable legal structure in order to commit to EGI.org in the long term;
4. mobilise national funding and resources and be able to commit to EGI.org financially, i.e. to pay EGI.org membership fees and, if there is a demand for such services in the NGI, request and pay for EGI.org services;
5. ensure the operation of a national e-Infrastructure to an agreed level of service and its integration into EGI;
6. support user communities providing general services to the applications and fostering the grid usage for new communities;
7. adhere to EGI policies and quality criteria.

ISSN: 2279-8773

The NGIs in Europe are currently at different levels of implementation, ranging from individuals claiming to represent an NGI, early implementations of NGIs with a preliminary legal status to fully government recognized legal entities. The project partners are listed in Table 2.2.

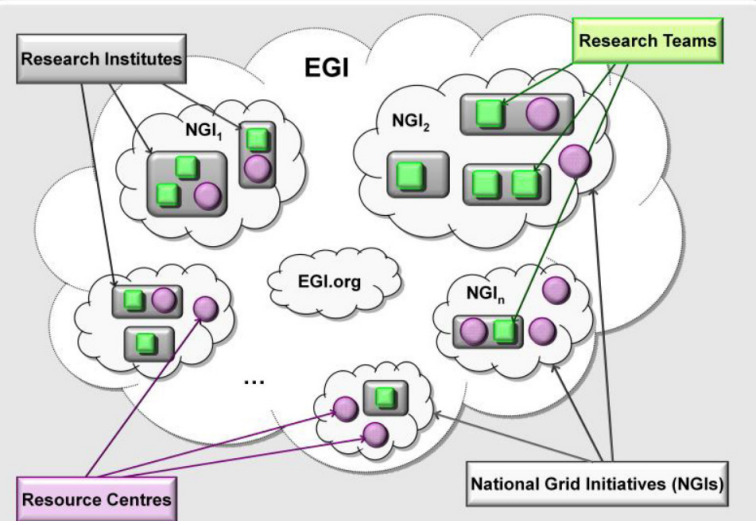

Figure 2.29: EGI and the NGIs

2.5.4 EGI Activities and user communities

As already mentioned, EGI is coordinated and managed by EGI.eu, a not-for-profit foundation established under Dutch law. In its coordinating role, and besides coordinating the infrastructure and support services, EGI.eu provides the following services to the user communities:

Operations management : The EGI.eu Operations Team coordinates and performs the activities required to deliver services at an agreed service level to the consumers of those services. Operations is also responsible for the ongoing management of technology deployment and technical support services.

Support and community coordination : EGI.eu coordinates support activities throughout EGI user communities and liaises with virtual research communities and virtual organisations to source their requirements.

Technology provision : All software upgrades and new programmes deployed on EGI are produced by independent technology providers. EGI.eu manages the outsourcing of technology developments, according to the requirements gathered by the User Community Board (UCB) and the Operations Management Board (OMB), negotiates with potential technology providers and assesses the new software quality.

Strategy, policy and collaborations : EGI.eu represents the members of the EGI federation in the wider Distributed Computing Infrastructures (DCI) community and champions the interests of its communities at the European Union level. The EGI.eu oversees the development of EGI strategy and policy documents and papers according to the wider European vision for research and innovation.

Community building and outreach : EGI.eu organises EGI flagship events the Community Forum in Spring and the Technical Forum in Autumn, as well as other smaller events and workshops throughout the year. EGI.eu communications team leads the community effort to promote the latest news and scientific achievements of the research communities using EGI.

A particular effort has been put in EGI to establish and maintain relationship among the different communities of users. To improve this collaboration a number of Virtual Research Communities (VRCs) were formed. VRCs are groups of like-minded individuals organised by discipline or computational model. VRCs typically have an established presence in their field (for example an ESFRI project, EIROFORUM laboratory or national research structure) and represent a well-defined scientific or research community. VRCs are self-organised research communities which give individuals within their community a clear mandate to represent the interests of their research field within the EGI ecosystem. They can include one or more Virtual Organisations. A brief description of the VRCs currently supported by EGI is given below.

High Energy Physics The Worldwide LHC Computing Grid (wLCG) is a global collaboration that links grid infrastructures and computer centres worldwide, set up to distribute, store and analyse the data generated by the Large Hadron Collider (LHC) experiments at CERN. wLCG is a mature research community, both in its own right and in its use of grid technology.

ISSN: 2279-8773

Hydro-Meteorology Hydro-Meteorology Research Community (HMRC) deals with problems involving the hydrologic cycle, the water budget, and the rainfall statistics of storms. The boundaries of hydrometeorology are not clear-cut, and the problems of the hydrometeorologist overlap with those of the climatologist, the hydrologist, the cloud physicist, and the weather forecaster.

The Life-Science Grid Community The Life-Science VRC covers notably the following scientific domains: bioinformatics, genomics, biobanking, medical imaging, (statistical) analysis and systems biology (e.g., virtual physiological human). It covers research groups from universities, research centers and industry, IT actors developing tools for Life Sciences, hospitals and ESFRIs.

Structural Biology Worldwide e-Infrastructure for NMR (WeNMR) brings together research teams in the structural biology and life science area into a virtual research community at a worldwide level, focusing on biomolecular Nuclear Magnetic Resonance (NMR) and Small Angle X-ray Scattering (SAXS). These research communities need virtual platforms to provide user-friendly computation tools supported by an underlying high performance e-Infrastructure.

Humanities Common Language Resources and Technology Infrastructure (CLARIN) is committed to establish an integrated and interoperable research infrastructure of language resources and its technology. It aims at lifting the current fragmentation, offering a stable, persistent, accessible and extendable infrastructure and therefore enabling eHumanities. Digital Research Infrastructure for the Arts and Humanities (DARIAH) was set up with the mission to enhance and support digitally-enabled research across the humanities and arts. DARIAH aims to develop and maintain an infrastructure in support of ICT-based research practices.

Similar agreements are currently being negotiated with other research communities in other fields of research like:

Astro(-particle) Physics The two major VOs in this domain, Planck and MAGIC, share problems of computation involving large-scale data acquisition, simulation, data storage, and data retrieval. The Planck satellite of the European Space Agency (ESA) will be launched in 2008 and aims to map the microwave sky with an unprecedented combination of sky and frequency coverage, accuracy, stability and sensitivity. The MAGIC application simulates

ISSN: 2279–8773

the behaviour of air showers in the atmosphere, originated by high energetic primary cosmic rays. These simulations are needed to analyze the data of the MAGIC telescope, located in the Canary Islands, to study the origin and the properties of high energy gamma rays.

Earth Science Research (ESR) Earth Science covers a large range of topics related to the solid earth, atmosphere, ocean and their interfaces as well as planet atmospheres and cores. Recently, members of the ESR Virtual Organization have worked on rapid earthquake analysis, helping the scientific community to better understand these devastating natural disasters.

Computational Chemistry The main VO in the field of computational chemistry is COMPCHEM, that is managed by the Dept. of Chemistry of the University of Perugia (IT) whose horserace is the set of programs being part of the *a priori* molecular simulator GEMS. Several Computational Chemistry applications have already been ported to the Grid and have been run in production to calculate observables for chemical reactions, to simulate the molecular dynamics of complex systems, and calculate the structure of molecules, molecular aggregates, liquids and solids.

2.6 User Community Software

To support user communities in their efforts to carry out research in science and innovation, EGI-InSPIRE has promoted the development of various tools, services, workflows and schedulers of which we list some of the most popular ones

2.6.1 Tools and fron ends

EDMS EDMS [85] (Experiment Dashboard Monitoring System), is a software monitoring, transferring data and site commissioning that provides also assistance and VO management that was originally designed to support LHC, CMS, ATLAS and ALICE experiments but can operate on several Grid middlewares to cover the full range of the needed computational activities. The goal of theEDMS is to monitor the activities of the VO supported experiments on the distributed infrastructure providing a uniform

ISSN: 2279-8773

and complete view of the various activities like job processing, data movement and publishing, access to distributed databases regardless of the underlying Grid flavor.

GANGA GANGA [86] is an easy-to-use frontend for job definition and management that provides a uniform interface across multiple distributed computing systems. Ganga is the main end-user distributed analysis tool in ATLAS and LHCb experiments, and is also used as the foundation layer for other services such as HammerCloud to manage large numbers of jobs (see HammerCloud statistics). Ganga is a widely used tool. From May 2010 to February 2011 Ganga was deployed in 127 sites and was used by 1316 users (40% ATLAS, 40% LHCb, 20% others) to submit more than 250,000 jobs weekly. It main work areas are:

1. Providing persistency solutions to serve all user communities in a scalable way.
2. Extension with a plug-in-structure for access to different computational resources.
3. Integration with monitoring services, notably Dashboards, by reusing common software elements in the presentation layers increasing the coherence of the look and feel of the tools for the end-users.

DIANE DIANE [87] is a lightweight task processing framework which allows for more efficient and robust execution of large numbers of computational tasks in unreliable and heterogeneous computing infrastructures. DIANE is an established software tool, used by power-users in several communities to implement complex and demanding task processing use-cases. The planning process involves the interested power-users and the main features are

1. Integration of application-level monitoring information with existing GANGA/DIANE Dashboard.
2. Worker node failure detection algorithms to make it easier for users to specify correct values and avoid deadlocks.

Both Ganga and DIANE are used by power-users in various communities both within and outside HEP for running large-scale computing tasks and the use of these tools has been reported by communities from more than 10 scientific fields and disciplines. From May 2010 to February 2011 DIANE was deployed in 23 sites and was used to process more than 600,000 tasks.

ISSN: 2279–8773

Besides the LHC communities, the use of Ganga/DIANE tools has been reported for UNOSAT applications, Geant4 medical and space simulations and grid-enabled regression testing, AvianFlu Drug Search, ITU digital broadcasting planning, LatticeQCD simulations, Fusion, Image Processing and Classification, EnviroGrids, Simulation of Gaseous Detectors (GARFIELD).

2.6.2 Services

HYDRA

Hydra [88] is a file encryption/decryption service developed as part of the gLite middleware. Hydra is a special secure metadata catalogue designed to hold encryption keys. The Hydra functionality is accessible in the regular gLite User Interface and Worker Nodes through command line tools. Hydra may be deployed as a single key store or as a distributed key store, implementing the Shamir's secret key sharing algorithm, for improved availability and higher robustness against attacks.

A Hydra catalogue has been deployed within the EGI-InSPIRE project as a service for the life sciences community. Hydra is mature software and the only planed developments for this services are bug fixes and updates to preserve the hydra functionality in the future versions of the gLite middleware.

GRelC

During the EGI-InSPIRE project, the GRelC [89] system (the network of GRelC services deployed within EGI) has been enhanced to support the EGI communities with a new set of functionalities. These will be accessed by end-users through the GRelC Portal, a seamless, ubiquitous and web-based environment for the management of geographically spread and heterogeneous grid data sources.

An important task is related to the monitoring and control functionalities of the GRelC system infrastructure. Such a management framework is available through the GRelC Portal by means of a new set of web pages exploiting the dashboard approach. The dashboard provides a unified view (including charts, reports, tables) about the GRelC deployment, the status of the services, the available grid-databases, the

ISSN: 2279–8773

supported VOs, etc. A small set of sensors are able to collect the needed information, storing them into a relational database (named system catalogue). The dashboard is designed and implemented taking into account the Web2.0 paradigm (e.g. an important feature is the capability to export in other web applications some/all views provided by the DashMon just with few HTML lines of code) and represents an important database-oriented tool providing effective views both in the large and in the small.

The roadmap for the GRelC service foresees the creation of the EGI Database of Databases, a registry service accessible through a web page that will contain all of the information about the grid-databases available in the GRelC System, the associated VOs, a short and a long description and some other useful metadata.

Through the registry, users should be able to:

- query the registry (exploiting a keyword-based approach) asking for specific databases, filtering by VO, keywords, domain, etc. This will help people working in a specific domain to quickly identify available and related resources, identify key people working on specific subjects, easily contact them to establish collaborations, etc.
- join a specific grid-database, submitting via the web a request to the grid-database administrator to know more about the supported VOs, etc.
- add comments on the available data and the related data sources, being part of a community exploiting a collaborative and Web2.0 oriented approach. All of this data will be available for future users, creating a knowledge base centered around community-oriented topics.

2.6.3 Workflows and Schedulers

The Grid workflow managers described here are simple and linear. In them an application just runs after reading the output of other applications, and in the more complete case imply the generation of the input for another application and the waiting of some input for the latter before finishing its execution.

ISSN: 2279–8773

Kepler

Kepler [90] is a free open source workflow application designed to help scientists, analysts, and computer programmers to create, execute, and share models and analyses across a broad range of scientific and engineering disciplines. Kepler can operate on data stored in a variety of formats, locally and over the internet, and is an effective environment for integrating disparate software components, such as merging "R" scripts with compiled "C" code, or facilitating remote, distributed execution of models. Using Keplerś graphical user interface, users simply select and then connect pertinent analytical components and data sources to create a scientific workflow, an executable representation of the steps required to generate results. The Kepler software helps users share and reuse data, workflows, and components developed by the scientific community to address common needs. The Kepler software is developed and maintained by the cross-project Kepler collaboration, which is led by a team consisting of several of the key institutions that originated the project: UC Davis, UC Santa Barbara, and UC San Diego.

P-GRADE

The Parallel Grid Run-time and Application Development Environment Portal [91] (P-GRADE Portal) is a web based, service rich environment for the development, execution and monitoring of workflows and workflow based parameter studies on various grid platforms. P-GRADE Portal hides low-level grid access mechanisms by high-level graphical interfaces, making even non grid expert users capable of defining and executing distributed applications on multi-institutional computing infrastructures. Workflows and workflow based parameter studies defined in the P-GRADE Portal are portable between grid platforms without learning new systems or re-engineering program code. Technology neutral interfaces and concepts of the P-GRADE Portal help users cope with the large variety of grid solutions. More than that, any P-GRADE Portal installation can access multiple grids simultaneously, which enables the easy distribution of complex applications on several platforms.

ISSN: 2279–8773

SOMA2

SOMA2 [92] is a versatile modelling environment for computational drug discovery and molecular modelling. SOMA2 is operated through a web browser. The SOMA2 environment offers a full scale modelling environment from inputting molecular data to visualisation and analysis of the results. The system makes use of the scientific applications installed in the computing system for which the SOMA2 environment includes interface tools. The scientific programs are easily configured and automatically executed in the SOMA2 environment. In addition, the SOMA2 environment offers a possibility to construct a computational workflow where several programs are run one after another. The SOMA2 environment is designed to be run so that the CGI web application is executed as an authenticated user to provide individualised and secure way of using the application. The SOMA2 modelling environment is developed and maintained by CSC. In earlier stages the SOMA2 R&D project was part of TEKES (National Technology Agency of Finland) funded DRUG2000 technology program. As of version 1.4 SOMA2 is enhanced to provide support to heavier grid computing. This development is funded by EGI (European Grid Infrastructure) foundation and is part of EGI-InSPIRE project.

TAVERNA

Taverna [93] is an open source domain independent Workflow Management System, a suite of tools used to design and execute scientific workflows. Taverna has been created by the myGrid project and funded through the OMII-UK. Taverna has guaranteed funding till 2014. The Taverna suite is written in Java and includes the Taverna Engine (used for enacting workflows) that powers both the Taverna Workbench (the desktop client application) and the Taverna Server (which allows remote execution of workflows). Taverna is also available as a Command Line Tool that for a quick execution of workflows from a terminal without the overheads of the GUI. Taverna allows for the automation of experimental methods through the use of a number of different (local or remote) services from a very diverse set of domains (biology, chemistry and medicine to music, meteorology and social sciences). Effectively, Taverna allows a scientist with limited computing background and limited technical resources and support to construct highly complex analyses over public and private data and computational resources, all from a standard PC, UNIX box or Apple computer.

ISSN: 2279–8773

2.6.4 Parallelization libraries: MPI

Execution of MPI applications requires sites that properly support the submission and execution of parallel applications and the availability of a MPI implementation. For such reason the MPI sub-task has been created to produce numerous MPI workbenches of increasing complexity with specific high impact on the Computational Chemistry and Fusion communities. These products will also have impact on other User Communities. The core sub-task objectives are:

- Improved end-user documentation, addressing MPI application development and job submission in ARC, gLite and UNICORE,
- Quality controlled MPI site deployment documentation,
- Outreach and dissemination at major EGI events and workshops,
- User community, NGI and site engagement, gathering direct input,
- Participation in selected standardisation bodies.

MPI sub-task partners have a great wealth of experience in designing, producing and deploying MPI applications under gLite. These range from relatively simple codes, to large scale production workflows using multiple externally provided (and widely used) MPI-enabled libraries. They also engage with the gLite, ARC and UNICORE communities producing high-level documentation for MPI application development and submission under these middleware. As part of User Community engagement effort, the MPI team regularly surveys Virtual Organisations, Users and Site administrators for critical feedback, acting as a means to gather information about current deficits and future requirements.

2.7 User application in MPI

The University of Perugia (UNIPG) cluster contributes to the activities of the MPI sub-task by making available applications (some of which are in-house designed, produced and deployed) under gLite middleware and investigating their parallel structure and their MPI implementation. UNIPG is a computer Science cluster made of researchers belonging to the Department of Chemistry, Department of Physics and the Department of Mathematics and Informatics of the University of

ISSN: 2279-8773

Perugia. The computational research activity of UNIPG is focused on codes of the area of Molecular and Materials Science and Technology (MMST) community. The following sections describe the related work performed within EGI-InSPIRE.

2.7.1 Linear algebra routines

A first study was carried out by considering the low level complexity codes taken from typical library routines or algorithms popular among the MMST community members. The work moved from preliminary attempts to implement of some linear algebra computation benchmarks on a model grid platform [94] The case dealt in some detail, here, is a set of three matrix multiplication algorithms:

- **Cannon algorithm:** [95] it was developed in 1969 and still represents a memory efficient version of the following matrix multiplication algorithm:

$$C_{i,j} = C_{ij} + \sum_{k=1}^{N} A_{ik} * B_{kj} \tag{2.4}$$

  This algorithm partitions A and B matrices into square sub-blocks. In particular, Cannon's algorithm make use of a mesh of s^2 processes that are connected as a torus. Process (i, j) at location (i, j) initially begins with submatrices $\mathsf{A}_{i,j}$ and $\mathsf{B}_{i,j}$. As the algorithm progresses, the submatrices are passed left and upwards, as sketched in Fig. 2.30:

  1. Initially $P_{i,j}$ begins with $\mathsf{A}_{i,j}$ and $\mathsf{B}_{i,j}$.
  2. Elements are moved from their initial positions to align them so that the correct submatrices are multiplied with one another. Note, please, that submatrices on the diagonal do not actually require alignment. Alignment is obtained by shifting the i-th row of A i positions left and the j-th column of B j positions up.
  3. Each process, $P_{i,j}$ multiplies its current submatrices and adds to a cumulative sum.
  4. The submatrices are shifted left and upwards.
  5. The above two steps are repeated through the remaining submatrices.

ISSN: 2279–8773

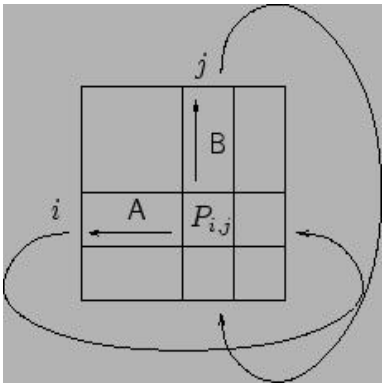

Figure 2.30: Scheme of the initial step of the Cannon algorithm

The parallel algorithm will be implemented by making use of the MPI1 and MPI2 libraries. In particular, a Master-Slave model of parallelism will be adopted in a typical SPMD approach. Moreover, due to the requirements of the algorithm, some features of the MPI library will be used, like the virtual topologies, the cartesian intra-communicators, and, depending on the platform, the RMA operations.

- **Fox algorithm:** The Fox algorithm [96] is similar to the Cannon one: in particular, it is a matrix multiply algorithm that uses a submatrix block cyclic data distribution. The communication pattern is asymmetrical: rows broadcast, columns rotate. Yet the Cannon algorithm performs uniform rotations. The Fox treatment makes the following assumptions:

  1. The number of processes (p) is a perfect square;
  2. The matrices to be multiplied are square of order n x n;
  3. sqrt(p) divides n evenly;

  The pseudo-code for the Fox algorithm reads:
  q = sqrt(p) // number of rows, cols in processor grid
  // A is operand 1, B is operand 2 in A * B
  // C is result
  // i,j = process row, column
  // src, dest rows for rotating 'up'
  src = i+1 mod q;
  dest = i-1 mod q;
  for (stage = 0; stage <q; stage++)
  k_bar = (i+stage) mod q;

broadcast(A[i,k_bar]) to row i;
C[i,j] = C[i,j] + A[i,k_bar]*B[k_bar,j]
sendrecv(B[k_bar,j],src,dest);

Also in this case, the parallel implementation will be similar to that of the Cannon algorithm. In particular, use of virtual topologies and communicator management primitives will be adopted.

- **Strassen algorithm:** The standard method of matrix multiplication of two n X n matrices takes O(n^3) operations. Strassens algorithm [97] is a Divide-and-Conquer algorithm that is asymptotically faster, i.e. O(n^{lg7}). The usual multiplication of two 2x2 matrices takes 8 multiplications and 4 additions. Strassen showed how two 2x2 matrices can be multiplied using only 7 multiplications and 18 additions. In particular, the Strassen algorithm multiplies two matrices, A and B, by partitioning the matrices and recursively forming the products of the submatrices. If we assume that A and B are n x n matrices and that n is a power of 2, if we partition A and B into four submatrices of equal size (2 x 2) and compute:
$P_1 = (A_{11} + A_{22})(B_{11} + B_{12})$
$P_2 = (A_{21} + A_{22})B_{11}$
$P_3 = A_{11}(B_{12} - B_{22})$
$P_4 = A_{22}(B_{21} - B_{11})$
$P_5 = (A_{11} + A_{12})B_{22}$
$P_6 = (A_{21} - A_{11})(B_{11} + B_{12})$
$P_7 = (A_{12} - A_{22})(B_{21} + B_{22}$
then it can be seen that:
$C_{11} = P_1 + P_4 - P_5 + P_7$
$C_{12} = P_3 + P_5$
$C_{21} = P_2 + P_4$
$C_{22} = P_1 + P_3 - P_2 + P_6$
If the conventional matrix multiplication algorithm is used, then there will be approximatively $7 \cdot 2(n/2)^3$ arithmetic operations in forming the P_i matrix products and $18 \cdot (n/2)^2$ operations in adding and subtracting the submatrices on the right hand side of the previous equations. If we adopt the Strassen algorithm, the number of arithmetic operations is reduced from $2n^3$ to $(7/8)2n^3$ in going from the conventional algorithm to the Strassen one.
In this case, the parallel implementation will be more complicated than in the previous ones. In fact, in terms of required commu-

ISSN: 2279–8773

nications the use of a divide-and-conquer parallel scheme is difficult to implement, and, in particular, the algorithm needs, from a computational point of view, an enormous number of recursive operations to reach the limit of the scalar x scalar multiplication. Therefore, this algorithm, in order to run faster than the standard one, will probably need some modifications in the parallel implementation.

2.7.2 Reactive scattering and Molecular Dynamics

The next level of complexity considered is that of Reactive scattering (ES) and Molecular Dynamics (MD) programs. For them we have operated both on in-house developed quantum RS codes (few atoms systems) and on popular MD packages (many particle systems). As to quantum RS codes the work has been concentrated from the very beginning on both fine and coarse grain MPI parallel scheme previously developed. Particular interest was paid to the total angular momentum (coarse) and to the wavefunction (or wavepacket) grid (fine grained representation in both the time dependent [98] and the time independent [99,100]) methods with application to the $N+N_2$ system [101]. Previously measured efficiencies and speedups for coarse and fine grained parallelism were confirmed by Grid calculations performed within the MPI studies of the EGI-InSPIRE project.

As to MD packages they are based on the application of the Newton second law $\mathbf{F} = m\ \mathbf{a}$, where $\mathbf{F}$ is the force exerted on the particle, m is its mass and $\mathbf{a}$ is its acceleration for all the particles of the considered system. The related set of equations (motion equations) allow to determine from the knowledge of the forces acting on every particle the corresponding acceleration. Integration of the equations of motion then yields a trajectory that describes the positions, velocities and accelerations of the particles as a function of time. From this trajectory, the average values of several properties can be determined by applying the ergodic hypothesis (over long periods of time, the time spent by a particle in some region of the phase space of microstates with the same energy is proportional to the volume of this region). The method is deterministic; once the positions and velocities of each atom are known at a given time, the state of the system can be predicted at any time in the future or the past. Molecular dynamics simulations can be time consuming and computationally expensive. The potential energy is a function of the atomic positions (3N) of all the atoms in the system.

ISSN: 2279–8773

Due to the complicated nature of this function, there is no analytical solution to the equations of motion; they must be integrated numerically. Different numerical algorithms have been developed for this, with the most popular being: Verlet algorithm, Leap-frog algorithm, Velocity Verlet and Beeman algorithm.

In the DL_POLY Molecular Dynamics code [11] both the Velocity Verlet and Leap-frog algorithm have been implemented. For our tests on the "valiomicine" we made use of the Leap-frog algorithm on the EGI resources that support COMPCHEM. The executable has been compiled statically in the UI (User Interface) machine used by COMPCHEM in order to assure binary compatibility. The compiler used was *ifort* (academic license) linked with MPICH2 libraries. From a preliminary analysis performed by running the *glite-wms-job-listmatch* and the related JDL file in which the requirements MPI-START & MPICH have been specified, resulted that 16 out of the 25 sites that support COMPCHEM support also MPI applications. The reduction in the number of sites supporting MPI from 22/25 in 2009 to 16/25 in 2010 is mainly due to the introduction of the MPI-SAM tests (now NAGIOS tests) which assure the basic requirements for a job submitted with MPI flags.

The performance of each site has been obtained running the DL_POLY executable sequentially in one node and in parallel in 2 ,4 ,8, 16, 32, 64 nodes on the same cluster, evaluating statistics and performances. The global performances and the statistical analysis carried out by submitting MPI jobs have been compared with those obtained in 2010 and in 2009.

As shown in Fig. 2.31 there is an overall improvement of the number of MPI jobs running correctly and we believe that there is still room for improvement, in particular in the right part of the graph where the CPU requirements are larger than 32.

Tabs. 2.3 and 2.4 show in a more quantitative way the percentages plotted in Fig. 2.31. That shows that abortion does still occur (probably due to a misconfiguration of some sites like hellasgrid.gr). This leads us to the conclusion that abortion is structural and that, for production purposes, this sites should be omitted from the pool of sites supporting MPI using the "Requirement" tag in the JDL.

During the tests a delay in the submission procedure has been registered and this could be mainly due to two factors:

- excessive load of the WMS during the submission procedure

ISSN: 2279–8773

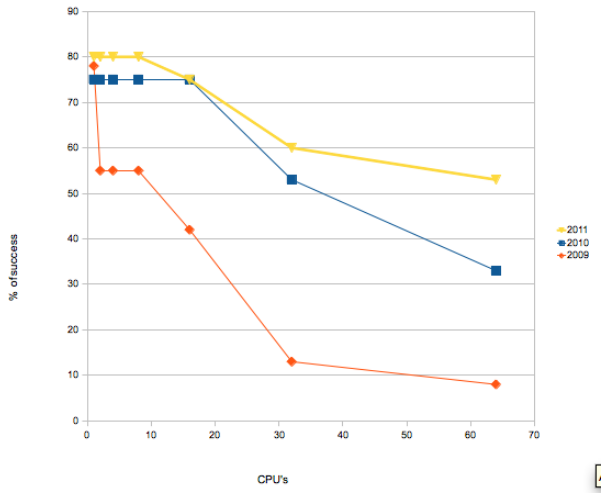

Figure 2.31: Visual representation of the statistical analysis plotting the percentage of successful jobs against the different years.

- number of CPUs required

In fact, as larger is the number of CPUs required for the calculation, longer is the match time to have all the CPUs free in the same cluster.

2.7.3 CHIMERE multi scale model

The highest level of complexity we tackled was the MPI structuring on the Grid of the package Chimere [102]. Chimere is the multiscale three dimensional Chemistry and Transport Model (CTM) package, owned by French institutes INERIS, LISA, CNRS, modeling the transformations of chemical species and the production of secondary pollutants in the atmosphere. Chimere, originally implemented on our machines in Perugia within a research agreement signed with the Regional Agency for the Environment (ARPA[10]) of the Umbria Region, is believed to be one of the modellistic packages better suited to deal with the chemistry nature of pollutants transformation like the physico-chemical processes concerned with diffusion, transport, deposition and photochemistry. It is in fact based on mechanisms combining a large family of chemical processes and the transport eulerian model. Chimere is designed to provide daily predictions of Ozone (O_3), sulphur oxides (SO_x), nitrogen oxides (NO_x), carbon monoxide (CO), all the volatile organic compounds but Methane (COVNM), particulate matter (PM_k

with k being the maximum value of the diameter of the particulate for the considered class) and other important atmospheric pollutants.

Implementation and parallel structure

The adopted version of Chimere (the V200606A one, written in FORTRAN 77 and later converted into FORTRAN 90, originally structured to run on x86 processors and Linux operating systems) has been installed on a cluster of 8 Intel nodes (Xeon 3GHz, 4GB ram, SLC5.7 operating system, Intel compiler, Gbit network) connected to the distributed computing Grid of EGI. This version of the package has a Multiple Program Multiple Data (MPMD) structure that is suited for use in MPI and needs in any case the running of at least two concurrent processes. Chimere is structured as a task farm concurrent model in which a main process (master) rules a certain number of processes (workers) working for it. The master sends to the workers the task to run and collects and stores the returned results. The method adopted by the developers to distribute the work among the workers is of the domain decomposition type. This method partitions the grid of the domain and assigns part of it to each process. The *nzonal* x *nmerid* bi-dimensional grid associated with each of the lowest eight layers of the troposphere (computing domain) is then partitioned into rectangular subdomains characterized by the two user defined variables *nzdoms* e *nmdoms* (with *nzdoms* indicating the number of subdomains in the direction west-east and *nmdoms* that in the direction south-north as shown in Fig. 2.32). The number of needed workers will be, therefore, *nzdoms* x *nmdoms*.

In order to effectively exploit concurrency and achieve significant computing throughput on the Grid a proper distribution model was adopted [103]. More in detail, this consists of an iterative structure of independent cycles to be executed a large number of times. Such a model exploits the fact that the dependence of the simulation relative to a given day does only partially (for a few hours) depend on the initial concentrations (i.e. the values of the previous day) since they rapidly converge to the actual solution regardless of the starting values. Accordingly, we have restructured Chimere so as to run concurrently several simulations for subsets of days the first of which replicates the last day calculations of another (arbitrary) subset. Then the results obtained using various subsets are glued together by discarding the starting day of each subset. The procedure was checked by comparing with an actual serial

ISSN: 2279–8773

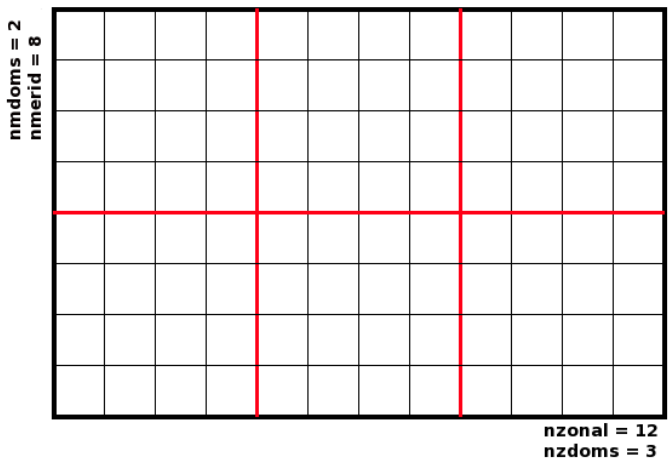

Figure 2.32: Subdomains division in Chimere

calculation by reproducing the heat wave that hit Europe in the period July 30 - August 3 of the year 2003 and no significant difference was found. The procedure is articulated as follows: first, the code is compiled using the Intel Fortran Compiler, the NetCDF libraries [104] with the support of the MPI libraries in order to maximize the performance of the multi processorWorking Nodes (WNs) of the segment of the production EGI grid available to the COMPCHEM VO. Second, the files necessary for the execution are uploaded to the Grid environment on one of the Storage Elements (a remote machine for the date storing that supports, via the gridftp protocol, the data transfer between the machines interconnected into the Grid) that supports the VO (in our case for example the se.grid.unipg.it, SE) before submitting the job. Third, the script is launched for execution.

Speed up and efficiecy

It has been found from the tests performed that the execution time of Chimere appreciably depends on the number of workers used. This seems quite reasonable due to the fact that the partitioning of the computational domain among an increasing number of workers makes the amount of work per processor decrease. To the end of measuring performances, *real time* (in seconds) values returned by the system command `time` at the end of the execution of `chimere.e` have been collected. Such a quantity indicates the net usage of the CPU inclusive

of the time necessary to the system to meet the requests of the process (system time).

To the end of evaluating the corresponding gain of time obtained when increasing the number of processes, the usually plotted quantity is the speedup (i.e. the ratio between the real time associated with the use of a single worker and the one associated with the use of a given number of workers). Such plot is shown in Fig. 2.33, in which the best (shortest) times for a given number of workers are plotted. The plot has an initial increasing trend starting from 1 (single worker) and keeps rising (though more smoothly) to reach a speedup close to 10 when 30 workers are used. Plotted values show large deviations from what can be extrapolated from the first ones. As a matter of fact, already with a number of workers slightly larger than 10, the deviation nears 100% with the speedup curve approaching a plateau (though still keeping a residual positive slope). Such behaviour is typical of parallel schemes requiring significant communication between the master and the workers that prevents additional gains. To verify the dependence of the execution time on the domain partitioning, the occupation of the workers has been analyzed confirming that in the combination 2x6 with 12 workers, for example, the last node is only half occupied. This means that only 2 processors out of 4 are engaged. In fact, on the first node there is always a processor engaged by the master task and of the remaining 3 processors one is left in occupied while the other two are engaged to run as workers. By following the criterion of maximum occupation, the other workers are 4 on the second node and 4 on the third node so that they are fully occupied. The remaining 2 workers run on the 4th node that results, therefore, half occupied. In the combination 2x5 with 10 workers, instead, the last node is totally occupied. In going from the 2x5 to the 2x6 combination the execution time increases suggesting that the best times are those in which the last node is fully occupied (as confirmed by the analysis of the other cases). To better single out such effect, the speedups achieved in the cases of a fully occupied or a partially occupied last node have been rationalized and shown in Fig. 2.34. In the graphs results corresponding to the same value of *nzdoms* or *nmdoms* has been connected obtaining, again, the typical trend in which the speedup smoothly increases before reaching a maximum and decreases again afterwards. The final decrease could be rationalized in terms of an increase of the time devoted to communications. As a matter of fact, when subdomains are created, the cells of the computational domain are partitioned by the subroutine of Chimere and seldom

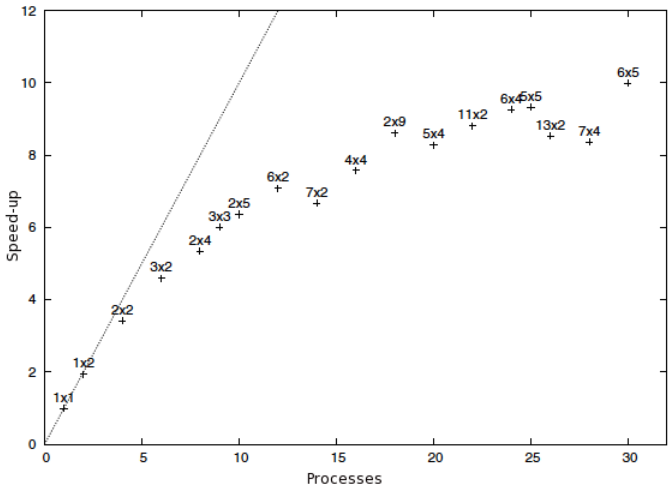

Figure 2.33: Best speed up value plotted as a function of the number of processes and the corresponding nzdoms x nmdoms combination of values are quoted above the symbol. The dotted line indicates the ideal trend

the cells are evenly divided among the subdomains. This makes the remainder of the division to be unevenly partitioned among the various subdomains.

2.7.4 GPU computing using a cloud approach

The importance of fine grained parallelism in the codes considered has motivated us to implement our codes also on GPUs.

GPU computing, or more in general the possibility of using the vector processors of graphics card as computational general purpose computing units has, in fact, generated considerable interest in the scientific community. In our case, the research effort has concentrated on the implementation of the already mentioned time dependent and time independent quantum reactive scattering codes of Refs [105–107]. However at present increasing emphasis is being put on Cloud Computing and more generally the opportunity to transparently use computational resources, together with the consolidation of virtualization technologies, allow to provide to the end users the needed specific environments for their activities. This growing interest for this two aspects has further motivated our research activity on how to use this technologies in a grid infrastructure. For such reason, in the work carried out by

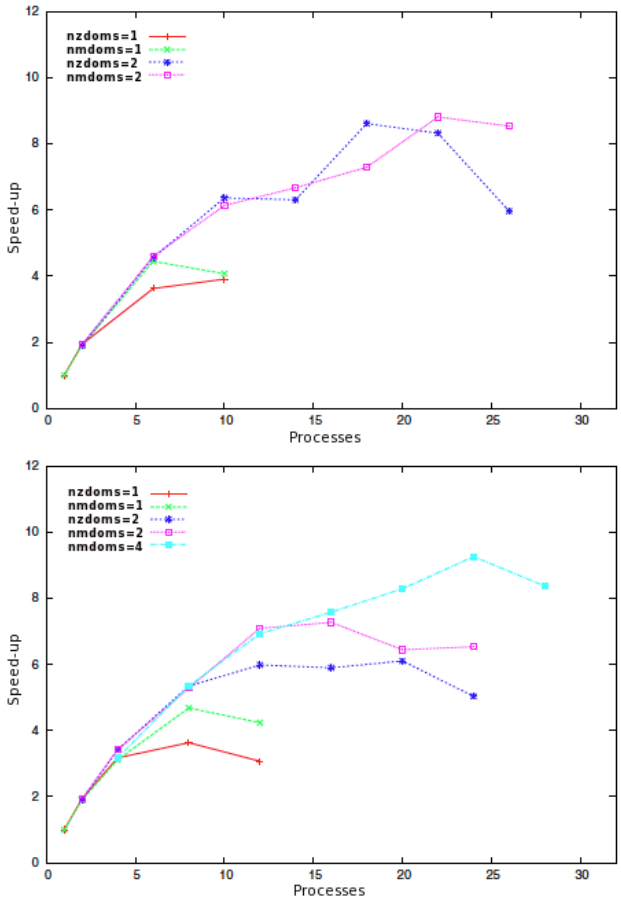

Figure 2.34: Plot of the speedup as a function of the number of processes when the last node is fully occupied (upper panel) and the last node is partially occupied (lower panel). In the two graphs constant nzdoms or nmdoms series have been evidenced

our group [108], we provided a on-demand GPU environment (GPU framework, Operating System and libraries) accessible via the EGI infrastructure making using a Cloud approach. The main purpose of the work was to provide a ready to use GPU environments for the communities using the EGI infrastructure to share GPU resources. As a single GPU environment does not satisfy the different requirements of these communities (such as operating systems, compilers and scientific libraries). For this reason, the developed system provides dynamical environments with the aim to optimize GPU resources usage. Contextually, the Cloud Computing opportunity allows to take into account

ISSN: 2279-8773

the GPUs as a Service (IaaS). A strategy to provide on-demand execution environments has been proposed through the joint usage of traditional and widespread gLite components and the popular standard EC2 [109] web-service APIs. An entire job flow that enables the Local Resource Management System (LRMS) to discriminate the GPU resources requests, through Glue Schema parameters, has been defined in order to allocate, in a dynamic fashion, the required resources on a Cloud-like infrastructure either public, private or hybrid. To achieve this goal, part of the work has been devoted to the virtualization of the physical GPU resources in order to make them available in a Infrastructure as a Service (IaaS) private Cloud [110–112]. To this end a centralized mechanism, responsible to listen for events generated by the LRMS like job scheduling and termination, has been implemented to keep track of each request. These events are then used to carry out the required actions as follows: once a job is received and identified as a GPU usage request, is treated as an event that triggers the allocation of virtualized resources according to simple leasing rules. In a similar way the termination of jobs are notified to a daemon that releases the execution environment. In order to develop and test the whole infrastructure, a fully working test bed has been built with the adoption of the Eucalyptus software [110] system to implement a private cloud over the cluster. We also addressed the need of the creation of Virtual Machine Images to match the requirements of the execution of GPU-dependent jobs, such as CUDA, OpenCL libraries and gLite middleware.

ISSN: 2279-8773

Table 2.2: EGI-InSPIRE project partners.

Code	Country	NGI
BE	Belgium	BEgrid - The Belgian Grid for Research
BG	Bulgaria	Bulgarian Grid Infrastructure
HR	Croatia	CRO NGI - Croatian National Grid Infrastructure
CY	Cyprus	Cyprus Grid Initiative
CZ	Czech Republic	MetaCentrum
DK	Denmark	TBA
EE	Estonia	Estonian Grid
FI	Finland	Finnish Grid Infrastructure
FR	France	France Grilles
DE	Germany	NGI-DE - National Grid Initiative for Germany
GR	Greece	GRNET - Greek Research and Technology Network
HU	Hungary	NGI-HU
IE	Ireland	Grid-Ireland
IL	Israel	ISRAGRID - Israel Nat. Infra. for Grid and Cloud Comp.
IT	Italy	IGI - Italian Grid Infrastructure
LV	Latvia	Latvian Grid
LT	Lithuania	LitGRID
MK	FYR Macedonia	MARGI - Macedonian Academ. Research Grid Initiative
MD	Moldova	MD-Grid - National Grid Initiative of Moldova
ME	Montenegro	MREN - Montenegro Grid Initiative
NL	The Netherlands	BIG Grid - The Dutch e-Science Grid
NO	Norway	NorGrid - Norwegian GRID Initiative
PL	Poland	PL-Grid - Polish National Grid Initiative
PT	Portugal	INGRID - Iniciativa National de Grid
RO	Romania	RoGrid - Romanian National Grid Initiative
RS	Serbia	AEGIS - Acad. and Educat. Grid Initiative of Serbia
SK	Slovakia	Slovak Grid
SI	Slovenia	SLING - Slovenian Initiative for National Grid
ES	Spain	NGI-ES - Spanish National Grid Initiative
SW	Sweden	SweGrid - The Swedish Grid Initiative
CH	Switzerland	SwiNG - Swiss National Grid Association
TR	Turkey	TRUBA - Turkish National Grid Initiative
UK	United Kingdom	UK NGI - UK National Grid Iniative

ISSN: 2279-8773

Table 2.3: Statistical analysis performed on the parallel jobs requiring up to 8 CPUs.

Job status	% (2009)	% (2010)	% (2011)
Success	53	75	100
Not success	47	25	0

Table 2.4: Statistical analysis performed on the parallel jobs requiring up to 16 CPUs.

Job status	% (2009)	% (2010)	% (2011)
Success	21	54	62
Not success	79	46	38

ISSN: 2279-8773

Virt&l-Comm.2.2012.4

ISSN: 2279-8773

Chapter 3

Simulation Workflows

3.1 Introduction

The assemblage and running of complex simulation made of several parts (often involving different know hows) is better managed through workflows. Workflows [1,2] are, in fact, specifically designed for dealing with data-intensive complex problems as is scientific research [3] especially when implemented on the grid for massive repeated executions. In particular Scientific Workflow Management Systems (SWMS) are a key technology aimed at integrate computing and data analysis components, and to control the execution and logical sequences among them. By hiding the management complexity in an underlying infrastructure, SWMSs facilitate scientists when designing complex scientific experiments, accessing geographically distributed data files, and executing the experiments using computing resources at multiple organizations. In this way, domain scientists can neglect inherent technicalities and effectively use available resources while focusing on the logic of the research problem.

Scientific workflows may have special features such as number crunching heavyness, data or transaction intensity, reduced human interaction, and a large number of jobs. In contrast with business workflows operating on rather small data sets and depending on complex control flows, SWMSs typically adopt a data-flow model suited to perform data intensive tasks [113]. In order to accommodate scientific workflows, an easy and efficient way of managing the components of simulations on the different available computing platforms is needed.

ISSN: 2279-8773

As a matter of facts most of the SWMS have been historically developed for applications in specific scientific domains, e.g., chemistry, bioinformatics, high energy physics, astronomy and so on. Related calculations are characterized, from the execution point of view, by the need for executing a large number of smaller independent jobs. This means that these domains strongly depend on computing throughput. Grid [114] promise indeed to be the platform to election for their elaborations. Grids have extended the number of resources available to e-scientists when using suitable SWMSs.Yet, they cannot achieve high performance execution providing fault tolerance and high resource utilization efficiency. Up to now, in fact, little work has been performed to integrate in SWMS High Performance Computing (HPC) requirements (such as low latency networks). despite the fact that HPC requirements are very common to many scientific applications and that these applications can benefit from the special features of SWMSs calling external services and accessing remote data in a way transparent to the final user. An important part in the workflow lifecycle [115] is represented by the user experience, as it has to support a kind of user-driven, incremental and prototypical approach [116]. Moreover, portal-based access to workflow systems proved to be beneficial in various scientific applications [117]. The structure of workflows is associated with a graph where each node in the graph is a processing element. While Directed Acyclic Graph (DAG) may contain sequence, parallelism and choice, non-DAG additionally include iteration structures. For many scientific applications, provenance of workflows for the data they derive and for their specification is crucial to permit result reproducibility, sharing, and knowledge re-use in the scientific community [118]. The application of semantic technologies for the management of scientific workflows is gaining importance to the end of further supporting the users to cope with the ever increasing number of already defined workflows, available components and results [119].

3.2 COMPCHEM: the Molecular Science Virtual Organization

An interesting use of the workflows is the one associated with the assemblage of the Grid Empowered Molecular Simulator (GEMS) [5] undertaken by COMPCHEM VO in collaboration with the Department of Mathematics and Informatics of the University of Perugia.

ISSN: 2279–8773

COMPCHEM [6] has been assembled by the group of Computational Dynamics and Kinetics of the Department of Chemistry of the University of Perugia (in which the work of the present thesis has been performed) by gathering together a group of European molecular and material sciences laboratories committed to implement their computer codes on the production EGI Grid infrastructure.

3.2.1 The structure of COMPCHEM

The COMPCHEM Virtual Organization offers to its members clear advantages for carrying out their computational campaigns (especially when they are so complex to not be feasible using other computing platforms) in return for taking the burden of carrying out the extra duties necessary to work within the distributed Grid environment. After all, the status of COMPCHEM member may imply further levels of involvement with respect to the entry level of the VO (passive user, first layer) that offers to the user the possibility of implementing a code at wish for personal use. The reason for having established this entry level (that has a limited period of validity) is to check the laboratories on their real willingness to operate on a Grid platform. Already at this level, in fact, several competences necessary to restructure the code to run in a distributed way by exploiting the advantages of using a Grid platform need to be acquired (active user, second layer). In return one gets, as already mentioned, the advantage of distributing the code on a much larger platform and, on top of that, easier interaction with the other users and codes of the VO.

As sketched in Table 3.1 one becomes real member of COMPCHEM only after committing him/ herself to open the code implemented on the Grid to a shared usage by the other members of the VO (passive software provider, third layer). This implies the validation of a stable version of the code and the assemblage of all the necessary GUIs for use by other researchers. Software providers can also provide basic services like software maintenance services and user support (active software provider, fourth layer).

The commitment to confer to the Grid additional hardware (especially for those suites of codes which need special devices) after a negotiation with the Management Committee (MC) of the VO about the relevance of such a commitment to the strategic choices of the Virtual Organization is that of the fifth layer (passive resource provider). Also the

ISSN: 2279-8773

hardware providers can provide basic services and user support (active resource provider, sixth layer). Obviously, the conferring of both software and hardware to COMPCHEM will take place gradually due to the time needed to validate the software and to gridify the machines yet during such time the user will increasingly become a true collaborator of the VO.

A collaborator of the VO will likely devote to it also other unshared resources (e.g. for development work). To become an effective member of the VO and acquire the status of "COMPCHEM stakeholder" (seventh layer) a user should place a specific application to the MC. While the user status has a limited time validity (after which a user may become either a paying customer and/or a paid supplier of services) the status of member has, in principle, no time limit (though its terms could be periodically revised). The stakeholder, in fact, should take care of maintaining the software and the local segment of Grid hardware (a particular attention is needed for the conferring of software, either commercial or not, with special constraints like the payment of fees because in this case commercial, legal and financial aspects are better dealt centrally).

The members of the VO are requested to be proactive in providing either their own work or attract financial resources specifically for the development of the VO. As to contributing to the VO by providing work this may occur under the form of participation to the management of the Grid, to the development of SWMS, etc. As to attracting financial resources VO members should elaborate joint applications for funding, research projects and even develop within the VO commercial services. However, the most important contribution to the sustainability of COMPCHEM that is requested to the stakeholders is a high dynamism in research and in the transfer of its outcomes into innovation and developments (R&D). This means that, ideally, all members of the VO should excell in basic and applied research and be ready to provide work to be rewarded in terms of credit by the VO.

3.2.2 The structure of GEMS

As already mentioned the COMPCHEM main computational application is the GEMS [5] workflow. For this purpose several programs have been ported to the Grid and have been run in production. Efforts are also underway to further enrich the GEMS workflow and port

ISSN: 2279-8773

Table 3.1: Levels of membership in COMPCHEM.

Membership level	Description
User	*Passive*: Run a program implemented by other members of the VO *Active*: Implement at least one program for personal use
SW provider	*Passive*: Implement at least one program for use by other members *Active*: Interactive management of the implemented program for cooperative usege
Grid deployer	*Passive*: Confer to the Grid infrastructure at least a small cluster of nodes *Active*: Operates above the minimal level as support for the Grid deployment and management
Stakeholder	Take part to the development and the management of the VO

additional applications to the EGI infrastructure and promote wider collaboration between the computational chemistry research groups.

The aim of GEMS is to provide a simulation environment to investigate in an *ab initio* fashion chemical processes and to construct from first principles physical observables like, for example, the measured intensity of the product in a crossed molecular beam experiment [120] to study reaction dynamics of complex chemical systems.

Accordingly, the design of GEMS has been founded on the following sections: INTERACTION, DYNAMICS and OBSERVABLES (see Fig. 3.1 for more details).

INTERACTION is the first section of GEMS and carries out the theoretical step of the calculations which determine the electronic structure of the system. The calculated energies (usually called potential energy values) can be used directly (on the fly) as soon as they are produced. Most often, instead, they are first performed for a large set of geometries of the system and then properly adjusted to reproduce some known molecular properties. Finally they are interpolated using appropriate functional forms. Though not explicitly quoted here as a section, the interpolation procedure represents a true section of the simulator called FITTING that creates an accurate easy to use analytical Potential Energy Surface (PES).

ISSN: 2279–8773

DYNAMICS is the second section of GEMS and carries out the theoretical calculations determining the dynamics of the nuclei of the system. Most frequently these calculations are based on classical mechanics approaches. Elective approaches should be instead the quantum ones. Yet, at present, they are feasible in a rigorous way, only for three and four atom systems because the related computational machinery is extremely heavy. Because of this the efficient implementation of the related computational procedures on the computing grid is an extremely active field of research.

OBSERVABLES is the final section of GEMS and carries out the necessary statistical and model treatments of the outcomes of the theoretical calculations to provide an a priory estimate of the monitored (measured) properties of the system. Obviously, theoretical calculations may be confined only to a mere geometrical analysis of the molecular system or to an exploration of the Potential Energy Surface (PES). In the most rigorous approaches both the calculation of the potential energy surface and the integration of nuclear dynamics equations are performed and use is made of suitable computational packages. The procedures to calculate the observable properties of the system are, instead, specifically designed for the final application of interest and are most often entirely handled by the final users.

3.2.3 Implemented applications in GEMS: Interaction and Fitting

There are plenty of packages that could be incorporated into the GEMS workflows. The most popular of them are those implementing quantum chemistry methods (see for example Ref. [121] for more details). Most include the Hartree-Fock (HF) and some post Hartree-Fock methods. They may also include density functional theory (DFT). In the years MOLPRO [122], DALTON [123] and GAMESS [124] have been ported on the Grid and inserted then in the GEMS workflow. Theory and manuals for these codes are largely available and we do not further comment on them exept for the fact that our most recent and systematic work on *ab initio* electronic structure packages porting on the Grid has focused on GAMESS-US. GAMESS-US [124] is a program for ab initio molecular quantum chemistry. Briefly, GAMESS can compute SCF wavefunctions ranging from RHF, ROHF, UHF, GVB, and MCSCF. Correlation corrections to these SCF wavefunctions include Configuration Interaction, second order perturbation Theory, and

ISSN: 2279–8773

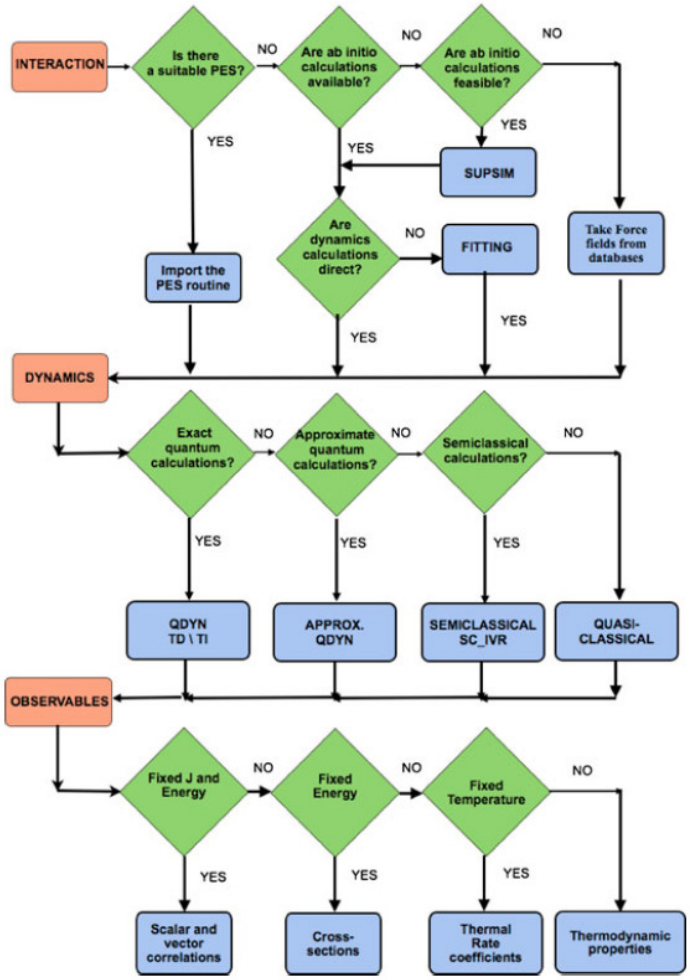

Figure 3.1: Detailed wokflow of the Grid Empowered Molecular simulator (GEMS)

Coupled-Cluster approaches, as well as the Density Functional Theory approximation. Geometry optimization, transition state searches, or reaction path following, vibrational frequencies with IR or Raman intensities and a variety of molecular properties, ranging from simple dipole moments to frequency dependent hyperpolarizabilities may be computed. Most computations can be performed using direct techniques, or in parallel on appropriate hardware. A detailed description of the program is available in Ref. [125]

More significant effort has been devoted to FITTING. FITTING is the

GEMS section working out the analytical representation of the ad hoc (either by purpose generated or taken from the literature) *ab initio* potential energy values by means of a suitable global functional form. Such global analytical representation, usually called Potential Energy Surface or PES, can be generated in different ways [126]. For example the three body (A, B, C) case as a many-body expansion [127] in the internuclear distances

$$\begin{aligned} V_{ABC}(r_{AB}, r_{AC}, r_{BC}) = & V_A^{(1)} + V_B^{(1)} + V_C^{(1)} \\ & + V_{AB}^{(2)}(r_{AB}) + V_{AC}^{(2)}(r_{AC}) + V_{BC}^{(2)}(r_{BC}) \\ & + V_{ABC}^{(3)}(r_{AB}, r_{AC}, r_{BC}). \end{aligned} \tag{3.1}$$

can be considered.

This means that the potential is written as a sum of three one-body terms ($V_A^{(1)}$, $V_B^{(1)}$, $V_B^{(1)}$), three two-body terms ($V_{AB}^{(2)}$, $V_{AC}^{(2)}$, $V_{BC}^{(2)}$), plus one tri-body term ($V_{ABC}^{(3)}$). The one-body terms are the energies of the separated bodies in their corresponding state (three constant values). The IJ two-body terms are potential energy curves depending on the related body-body distance (which include the nuclear repulsion) and are usually expressed as low order polynomials either of the inverse or of the damped distances r_{IJ}. As this is the case of the GFIT3C routine used in the present work [7], they can also be expressed as a low order polynomial of the bond order variables ($n = exp[-\beta(r_{IJ} - r_{IJe})]$), in which r_{IJe} is the two-body equilibrium distance, or of mixed variables like $\xi_{IJ} = r_{IJ}exp(-\beta_{IJ}r_{IJ})$ in which ξ_{IJ} is optimized to represent the long-range term of the two-body potential. The three-body term $V_{ABC}^{(3)}$ corresponds to the residual interaction due exclusively to three body forces. The advantage of this development is that, if we impose certain restrictions on the inter-body distances, the correct asymptotic limits are obtained. The value of the three body residual term is easily obtained by subtracting the values of the one and two body potentials to the (if necessary adjusted to reproduce the measured spectroscopic information of stable intermediates and fragments) calculated *ab initio* values. The three body term is also formulated as a polynomial in the above mentioned variables (let us call them basis functions $b_k(\mathbf{r})$ with $\mathbf{r}$ being the set of coordinates adopted to describe the potential) and a_k the related coefficients.

To the end of calculating the values of the a_k coefficients of the adopted basis best fitting the (adjusted) *ab initio* values the following equation

ISSN: 2279-8773

$$V_{ABC}(\mathbf{r}) = \mathbf{a}^T \mathbf{b}(\mathbf{r}) = \sum_{k=1}^{K} a_k b_k(\mathbf{r}) \tag{3.2}$$

(where the superscript T means transpose and K the number of basis functions) has to be solved. By interpolating the coordinates and the potential energy values $\mathbf{v}(j)(j = 1, 2, ..., J)$ of the J calculated points $\mathbf{r}(j)$ in order to determine the coefficients a_k we consider the functional $E(V)$ defined as

$$E(V) = \sum_{j=1}^{J} w_j [V(\mathbf{r}_{(j)}) - \mathbf{v}(j)]^2 \tag{3.3}$$

and impose it to be stationary. This implies the solution of the following "normal" equations

$$BWB^T \mathbf{a} = BW\mathbf{v} \tag{3.4}$$

in which W is a diagonal matrix made of all weights

$$W = diag[w_1, w_2, ..., w_J] \tag{3.5}$$

and B the matrix of values made of the basis functions $b(\mathbf{R})$.

3.2.4 Implemented applications in GEMS: quantum methods

The most important and original contribution of the present thesis to the development of GEMS is the one related to quantum reactive scattering applications of three body (A+BC) systems on the DYNAMICS section. In particular the applications considered deal with the calculation of the fixed total angular momentum quantum number J and the exact quantum three-dimensional state-to-state partial probability $P^J_{vjv'j'}(E)$, of the three body process

$$\mathrm{A} + \mathrm{BC}(v, j) \rightarrow \mathrm{AB}(v\prime, j\prime) + C \tag{3.6}$$

in which v and j label the internal energetic states of the initial two body cluster (unprimed quantities) while the corresponding v' and j'

ISSN: 2279-8773

(primed quantities) label the internal energetic states of the final two body cluster. The $P^J_{vjv'j'}(E)$ values calculated at a given value of the total energy E and of the total angular momentum quantum number J are then combined to estimate some observable quantities of the considered process. For this purpose $P^J_{vjv'j'}(E)$ is formulated in terms of the square modulus of the corresponding detailed **S** matrix elements $S^J_{vjKv'j'K'}(E)$ as follows:

$$P^J_{vjv'j'}(E) = \frac{1}{(2K_{\max}+1)} \sum_{K=-K_{\max}}^{K_{\max}} \sum_{K'=-K'_{\max}}^{K'_{\max}} \sum_{p=0}^{1} \left| S^{Jp}_{vjKv'j'K'}(E) \right|^2 \tag{3.7}$$

where $K_{\max} = \min(j, J)$, K is the helicity quantum number (the discrete body-fixed projection of the total angular momentum **J**) and p is the total parity. The state-specific partial probabilities, $P^J_{vj}(E)$, can be then calculated by summing over v' and j':

$$P^J_{vj}(E) = \sum_{v'} \sum_{j'} P^J_{vjv'j'}(E) \tag{3.8}$$

From the state-specific partial probability the state-specific cross section, $\sigma_{vj}(E)$, is derived using the following expression [128]:

$$\sigma_{vj}(E) = \sum_{J=0}^{J_{\max}} \sigma^J_{vj}(E) = \sum_{J=0}^{J_{\max}} \frac{\pi}{k^2_{vj}} \frac{(2J+1)}{(2j+1)} (2K_{\max}+1) P^J_{vj}(E) \tag{3.9}$$

where $\sigma^J_{vj}(E)$ is the state-specific partial cross section with $k^2_{vj} = 2\mu E_{\rm tr}/\hbar^2$ (k is the wavenumber of the system in the vj state) and μ being the reduced mass of the initial configuration. Converged state-specific cross sections require the sum of all the contributing J values. By carrying out the calculation for all the populated initial states at a given temperature T, one can evaluate the thermally averaged cross section $\sigma(E_{\rm tr})$ the thermal rate coefficient $k(T)$.

Detailed **S** matrix elements are calculated by integrating the Time Independent (TI or stationary) Schrödinger equation

$$\hat{H}\psi = E\psi \tag{3.10}$$

ISSN: 2279–8773

with $\hat{H}$ being the system Hamiltonian and ψ the time independent component of the nuclei.

For the present work the $S^J_{vjKv'j'K'}(E)$ values are calculated using a hyperspherical formalism in which the hyperradius $\rho = \sqrt{R^2 + r^2} = \sqrt{R'^2 + r'^2}$ is taken as the continuity variable of the related differential equations and the internal component of ψ is expanded locally (at fixed ρ values called sectors) in terms of the Delves basis functions $B^{JM}_{vjK}(\alpha, \theta)$ (α and θ are the Delves hyperangles). The matrix of the coefficients of the expansion $\mathbf{g}(\rho)$ is then propagated from the strong interaction region ($\rho = 0$) to the asymptotes (large ρ values) by integrating the following differential equations

$$\frac{\mathrm{d}^2\mathbf{g}}{\mathrm{d}\rho^2} = \mathbf{O}^{-1}\mathbf{U}\mathbf{g} \tag{3.11}$$

in which

$$O^{v'j'K'}_{vjK} = \langle B^{JM}_{vjK} | B^{JM}_{v'j'K'} \rangle \tag{3.12}$$

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

$$U^{v'j'K'}_{vjK} = \langle B^{JM}_{vjK} | \frac{2\mu}{\hbar^2}(\bar{H} - E) - \frac{1}{4\rho^2} | B^{JM}_{v'j'K'} \rangle \ , \tag{3.13}$$

while μ is the reduced mass of the system and $\bar{H}$ the assemblage of all the terms of the full Hamiltonian which do not contain derivatives with respect to ρ. The $\mathbf{S}$ matrix elements are evaluated by mapping the value of the wavefunction calculated at the asymptotes onto the product states.

Conceptually simpler is the Time Dependent (TD) approach in which the application of the hamiltonian to the wavefunction of the nuclei is related to its time derivative. Relevant codes considered for implementation on the Grid are:

- **RWAVEPR** [129] it integrates rigorously the three-dimensional time-dependent Schrödinger equation for a generic atom-diatom reaction by propagating wave packets. It calculates the scattering S matrix elements for given values of the vibrational quantum number, the rotational quantum number, the total angular quantum number, the quantum number for the projection of the total

ISSN: 2279-8773

angular momentum on the atom-diatom vector, for a given the parity and for a given range of total energies. From the **S** matrix elements the state-to-state reaction probabilities are calculated. The centrifugal sudden approximation (i.e. to neglect the Coriolis coupling) can be also invoked.

- **ABC** [8] is a program that uses a coupled-channel hyperspherical coordinate method to solve the TI Schrödinger equation for the motion of the three nuclei (A, B, and C) on a single Born-Oppenheimer potential energy surface. The coupled-channel method used involves a simultaneous expansion of the wavefunction in the Delves hyperspherical coordinates of all three chemical arrangements (A+BC, B+CA, C+AB). The quantum reactive scattering boundary conditions are applied exactly at the asymtotes potential, and the coupling between orbital and rotational angular momenta is also implemented correctly for each value of the total angular momentum quantum number.
- **MCTDH** [130] is a program implementing the MultiConfigurational Time-Dependent Hartree (MCTDH) method is nowadays considered as one of the most powerful tools for the quantum dynamics simulation of multidimensional systems. Unlike conventional wave packets methods, in the MCTDH approach the wave function is expressed on a basis of time-dependent functions, which evolve along with the system. The use of this time-dependent basis set turns up into a much smaller basis dimension and thus a greater computational efficiency with respect to standard wave packet approaches.
- **FLUSS** [131] The fluss code performes a modified Lanczos iterative diagonalisation of the thermal flux operator. The output of the code is a set of eigenvalues and eigenstates which can afterwards be used to calculate the thermal rate constant of a chemical reaction. A Krylov space is generated by recursive application of the thermal flux operator onto an initial wave function, typically a Gaussian-type wave packet located in the vicinity of the transition state. The matrix representation of the operator in the Krylov-type basis is diagonalized to obtain eigenstates and eigenvalues.

Among the implemented codes also the semiclassical (SC) initial value representation (IVR) program [132] has been implemented. SC-IVR method calculates the thermal rate coefficients for atom-diatom systems. The program adopts Cartesian coordinates in the full space. It

ISSN: 2279–8773

does not invokes the conservation of total angular momentum J to reduce the problem to fewer degrees of freedom and solve the problem separately for each value of J, as is customary in quantum mechanical treatments. The code uses the semiclassical coherent-state propagator of Herman Kluk (HK), the Bolzmannized flux operator is tuned continuosly between the traditional half-split and Kubo forms and the normalization integral is expressed in terms of simple constrained partition functions.

3.2.5 Implemented applications in GEMS: classical methods

The second largest class of established MMST programs are the MM (Molecular mechanics) and the Molecular Dynamics (MD) ones in which the atoms and molecules are allowed to interact and move according to classical mechanics equations of motion (though in GEMS using inhouse applications also quantum mechanics equations for small systems). Some of them have been already quoted in Ref. [121]. The popular packages implemented in Grid by us are:

- **VENUS** [133] calculates the cross-sections and rate coefficients for elementary chemical reactions by simulating the collisions between atoms and molecules whose initial conditions are sampled using a Monte Carlo scheme. This application is a modified version of the VENUS96 program by W.L.Hase (QCPE-671). It calculates the trajectory for two reactants (atoms or molecules) by integrating the Hamilton equation in cartesian coordinates. Before the collision the molecules are selected at discrete internal energy states and after the collision a quantization of the internal energy is also enforced on the product molecule. Initial positions and momenta are selected by using a Monte Carlo method. A parallelized version using MPI has been also implemented.
- **DL_POLY** [11] is a package of subroutines, programs and data files, designed to facilitate molecular dynamics simulations of macromolecules, polymers, ionic systems, solutions and other molecular systems on a distributed memory parallel computer. The package was written to support the UK project CCP5 by Bill Smith and Tim Forester under grants from the Engineering and Physical Sciences Research Council and is the property of the Science and Technology Facilities Council (STFC).

ISSN: 2279-8773

- **GROMACS** [10] (a MD simulator primarily designed for biochemical molecules, like proteins and lipids that have a lot of complicated bonded interactions, that is extremely fast at calculating the nonbonded interactions (that usually dominate simulations) and is equipped with tools for input assemblage and output analysis.

3.3 The service oriented approach

Significant efforts have been paid by COMPCHEM to implement its layered structure (users, software providers, grid deployers, stakeholders) aimed at enhancing the cooperative services of the VO.

To this end COMPCHEM is developing tools to provide its user with instruments for an optimized usage of the hardware and software of the Grid along the line of an effective Service Oriented Architecture (SOA) methodology [134]. In this spirit Grid empowered versions of the simulators SIMBEX [135] and GEMS [5] developed by some members of the community for the needs of crossed beam studies have been implemented. More recently this has prompted also the development of appropriate frameworks (like GriF [136]) allowing to define event-related dependencies between complementary applications (which may involve different computer environments) in a workflow-like fashion. In progress are also the activities carried out in cooperation with the laboratories of the former COST Action D37 called Grid Computing in Chemistry: GRIDCHEM [137] and the COST Action CM1002 called COnvergent Distributed Environment for Computational Spectroscopy (CODECS) [138] to design and develop a collaborative grid empowered simulator for Spectroscopy. A similar effort is being paid by the members of COMPCHEM to design and develop a collaborative grid empowered simulator for Cleaner Combustion within the activities of the COST Action CM901 called Detailed Chemical Kinetic Models for Cleaner Combustion [139]. Finally in a cooperative endeavour is also being carried out within the activities of the WeNMR VO [140] to design a simulator for NMR studies.

3.3.1 A first attempt to build a workflow

By making reference to the just mentioned grid activities it is important to describe how the SOA approach has been implemented. A

ISSN: 2279–8773

fundamental support to SOA collaborative activities is given by workflows [141] which facilitate the composition and coordination of different programs in a structured and flexible execution scheme. Such work is aimed at developing high throughput workflows by taking some specific complex computational applications as use cases.

A first attempt to build a suitable workflow for the TI calculations was carried out in ref [142] by writing the proper procedure needed to manage the flowchart of GEMS. This made it possible to calculate single E, single J S matrix elements for some atom-diatom processes. A general scheme for the resulting concurrent organization of the GEMS related computer programs on the Grid is the following: a distribution procedure iterates over the E, J pairs to perform the recursive TI integration of eq. 3.11 (an equivalent TD integration is performed integrating on time over initial conditions).

In this first workflow the computation is articulated as a coarse grained uncoupled loop made possible by the moderate energy values considered and, ultimately, by some adiabatic or energy shifting [143] approximation. The resulting distributed execution model is again typical of the "parameter sweeping" type.

The developed procedure is able to handle large sets of jobs. Each job execution requires the sending to the Grid of an execution script, of a specific input file and of the related program. The execution script is the same for all jobs while the input file is different for each job. In order to better cope with the heterogeneous nature of both the computing hardware and software (compilers, libraries, submission systems, etc.) of the Grid, executable rather than source programs were distributed over the net. In fact, in spite of the fact that the time required for sending the source code is considerably shorter than that required for sending its executable (moreover the sending of the executable is also more selective in terms of the type of adoptable machines) this approach exploits the fact that there is no need for identifying the machine compiler, selecting the optimal compilation options, compiling the code and verifying that all runs give exactly the same results as the ones obtained on the original machine.

Unfortunately, the software dependency of some computational codes from specific libraries and compilers, seldom available in the most common Linux distributions over the grid, make this approach not always viable and the execution of the code became strictly dependent from the machines where the calculation is performed.

ISSN: 2279-8773

3.3.2 An advanced Grid-based workflow model

To improve on the first version of the workflow, use has been made of the high throughput execution framework GC3Pie [9] and of the AppPot [144] cloud/grid virtual machines (both developed by the GC3 group). The specific complex computational application considered was the implementation of GFIT3C to be used for a quantum mechanical application being part of the Simulator.

In our work, the Python-based GC3Pie framework has been modelled to the needs of the previously quoted GFIT3C and ABC computational codes in order to build a flexible workflow.

GC3Pie is a library of Python classes designed for running large job campaigns (high throughput) on diverse batch-oriented execution environments, including ARC-based computational grids [73]. It also provides facilities for implementing command-line driver scripts, in the form of Python object classes whose behavior can be customized by overriding specified object methods.

At the heart of the GC3Pie model is a generic *Application* object, that provides a high-level description of a computational job: list of input/output files, what command to run, resource requirements and limits, etc. GC3Pie translates this information into the job description format needed by the actual execution back-end selected, e.g., xRSL for ARC-based Grids, or a submission script for direct execution on a batch-queuing system. *Application* objects can be adapted to provide behavior customized to a specific use case as for the AppPot Virtual Machine approach described later.

Workflow composition

GC3Pie provides composition operators, that allow treating a collection of tasks as a single whole [145]: compositions operators can be arbitrarily nested, allowing the creation of complex workflows, including workflows with cycles in them, and dynamic workflows whose structure is created at runtime by the script itself. Two base composition operators are provided, upon which others can be created (by subclassing).

The *ParallelTaskCollection* operator takes a list of tasks and creates a task that executes them all independently and, compatibly with the allowed degree of job parallelism, concurrently.

ISSN: 2279–8773

The *SequentialTaskCollection* operator takes a list of tasks and creates a task that executes them in the sequence; a decision method is invoked in between each step, which can terminate execution early (e.g., in case of errors), but also alter the list of planned tasks.

See section 3.3.3 for an overview of how these composition operators are applied to the GFIT3C+ABC workflow.

Command-line scripts

GC3Pie provides template command-line scripts for frequently-occurring use cases, in the form of reusable Python objects. The most used command-line pattern, and also the one we chose for the GFIT3C+ABC workflow, is called a *SessionBasedScript*. This template script manages a collection ("session") of tasks (where each task can be itself a collection, see above); tasks in the session are submitted to the grid and monitored until they have ended execution; the output from completed tasks is retrieved and stored in a configurable location. The *SessionBasedScript* also ensures the session is persisted to disk and correctly restored in case the program is interrupted or crashes and restarted later.

The AppPot Virtual Machine

AppPot [144] is a software system comprising a standard Linux Virtual Machine (VM) image and a set of auxiliary programs that make the AppPot software self-contained so that it can be deployed by just copying a few files. AppPot provides a way to run commands inside the VM, possibly in a non-interactive fashion, and to copy files in and out of the VM filesystem.

AppPot is based on the User-Mode Linux [146,147] virtualization technology. User-Mode Linux (UML) provides a way to run a Linux VM inside a Linux host using only a code that runs in "user space" and thus requires no "root" privileges or any other form of systems administrator support or consent. Only two files are needed for running a UML Virtual Machine: the UML kernel and the VM disk image. The UML kernel is a single executable file consisting basically of a regular Linux kernel modified to run as a process in a Linux host. It is possible to compile the UML kernel statically, so that it can run on a wide spectrum of Linux systems.

ISSN: 2279–8773

Combining the features above, one can run Linux-based VMs as grid jobs and implement user-initiated, generic application deployment on a computational grid. Indeed, users can install a new application into a copy of an AppPot VM, and then package the modified VM as a job's input data; the job control script can make use of the `apppot-start` script to run any command inside the AppPot VM.

3.3.3 Workflow description

The practical motivation for implementing the GC3Pie workflow was offered by the need for running large ABC analysis campaigns [148]. In the description given below we start from a set of given input *ab initio* points of the PES and we build the ABC binary and execute it.

More in detail the steps involved in the considered workflow are the following:

(A) The *ab initio* points of the PES are passed as input to GFIT3C which produces a Fortran file ("fitting file") containing the PES routine for the considered system.

(B) The fitting file is then merged into the ABC source code, which is compiled into an executable binary.

(C) The resulting ABC binary is distributed and executed.

To develop the related workflow we considered in our work three different use cases:

(1) The user provides the *ab initio* points of the Potential Energy Surface; this executes steps (A), (B), and (C).

(2) The user provides the *fitting file* (produced by a previous run of GFIT3C or by any other means); only steps (B) and (C) need to be executed.

(3) The user provides his/her version of the ABC executable for distribution in a parameter study fashion: only step (C) is executed.

Out of the three different use cases, of which a graphical illustration is provided in Fig. 3.2, the first one is the most complete. For its porting, in fact, different factors needed to be taken into account like the dependencies from libraries and compilers used. As it is (i.e. without undertaking a restructuring and a debugging of the code) the ABC code, for example, is only compiled correctly by using the G95 Fortran

ISSN: 2279–8773

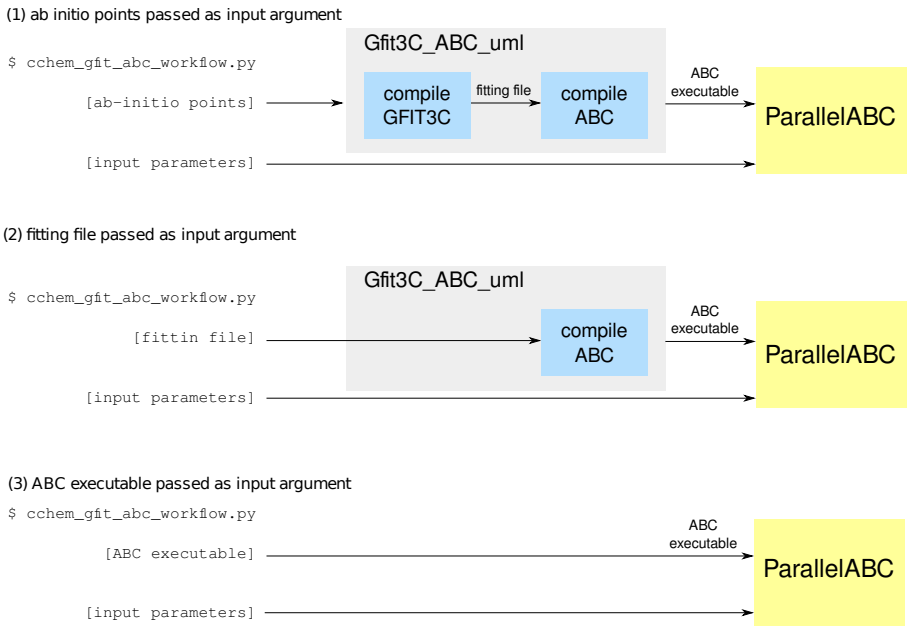

Figure 3.2: Sketch of the developed workflow showing its components and the connections between them.

compiler [149]. As this compiler is not natively available in the most common Linux distributions, step (B) cannot be in general executed on the computational nodes available on a grid infrastructure. A traditional approach to this problem would require users to negotiate the installation of the dependent software on a substantial fraction of the execution nodes. A simpler solution to this, however, comes from a VM-based approach: workflow steps are run within a Virtual Machine that provides a controlled and uniform execution environment, including all software dependencies. The AppPot software runs a Linux VM as a Grid job, allowing to roll out the planned solution on a Grid infrastructure without prior negotiations with Grid systems administrators. Moreover, the ABC code is rarely evaluated for a single set of parameters: in step (C) and, in general, in any parameter-sweep study, the program must be executed several times, consuming a large amount of CPU time. This justifies the use of an "embarassingly parallel" style of execution, in which many instances of ABC run independently and results are collected at the end.

```
$ cchem_gfit_abc_workflow.py --help
usage: cchem_gfit_abc_workflow [OPTIONS] [INPUT]

OPTIONS:
[...]
--g3c G3CFILE         GFIT3C input file.
--dim DIMENSIONS      Surface file.
--pes FORTRAN_PES     Fortran file for the potential energy surface.
```

Figure 3.3: Extract from the help text of the workflow driver script. Depending on the input provided by the user, a specific workflow task is created and executed.

Implemented application classes

The workflow driver has been implemented as a command-line script using GC3Pie *SessionBasedScript* template (an abstract of the script invocation syntax and help text can be seen in Fig. 3.3). The script searches for ABC input files by scanning an input directory (provided by the user in the command-line) and for each combination of ABC input files and *ab initio* points or fitting file (depending on command-line options), a workflow task is created and executed through completion.

Following the GC3Pie development model, most of the coding effort consisted in creating appropriate classes for specifying the workflow logic and the relation between the parts (see Fig. 3.2 for details).

The *ABC_Workflow* class represents the top level logic of the workflow: it conditionally enables the *MainSequentialABC* class (responsible for the source compiling stage) or the *ParallelABC* class (responsible for the ABC execution stage) for every input file and parameter combination provided by the user through the command line interface, defining the sequence of tasks to be executed.

If a compiling stage is needed, the *MainSequentialABC* creates an instance to the *Gfit3C_ABC_uml_Application* class (a *SequentialTaskCollection* according to the GC3Pie nomenclature) responsible for controlling steps (A) and (B). The script is able to build an AppPot VM, with the required software environment, in any of the execution nodes available on the Grid infrastructure and compile the source code of both GFIT3C and ABC (or only ABC depending on the provided parameters). At the end of this step, the statically compiled ABC executable and the related log files are transferred back to the user account in the client machine and are made available for a control check.

The management of single and multiple ABC instances are delegated to

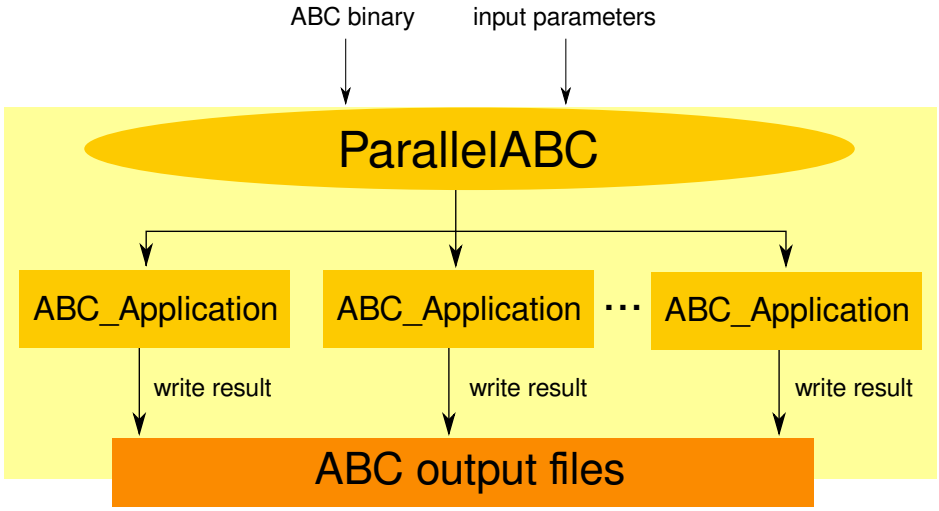

Figure 3.4: Sketch of the Workflow portion developed to distribute the ABC code into a grid environment making use of the GC3Pie framework.

different *Application* classes called *ABC_Application* and *ParallelABC* (a *SequentialTaskCollection* and a *ParallelTaskCollection* class, respectively, according to the GC3Pie nomenclature). The first class makes use of the GC3Pie libraries to submit and control the ABC execution as a single job in a grid environment. The second class, instead, manages the submission of multiple instances of the first class according to the number of input files provided by the user (see Fig. 3.4 for graphical details).

Also in this case the results produced by each computation, together with the file logs, are saved on the client machine. Attention has been paid to the case in which different output files must be stored in different directories to facilitate checks and further utilization by the end-user. Alternately, GC3Pie can leverage some features of ARC and upload the output files to a specified Storage Element (ARC supports the GridFTP and SRM protocols). In this case, output files are not donwloaded to the local directory: GC3Pie specifies an output location in the job description and the ARC server will upload the output files to the specified SE when the job is finished.

3.3.4 Benchmark usecases

To the end of testing the computational machinery and the validity of the developed workflow, we executed on the Swiss Multi-Science Com-

putational Grid (SMSCG) infrastructure [150] the complete workflow to deal with the exchange process

$$H_A + H_B H_C(v, j) \rightarrow H_A H_B(v', j') + H_C. \tag{3.14}$$

In the test we imported the *ab initio* points (as potential energy values) of Ref. [151]. In this approach,[1]therefore, the first step of the workflow is devoted to run the GFIT3C routine to the fitting of the *ab initio* points. From the fit a r.m.s. of 0.19 kcal/mol was obtained using a polynomial of degree 7 for the two body terms and a polynomial of degree 10 for the three body one. These values are completely in line with those reported in Ref. [151].

In the second step of the workflow the PES Fortran routine, obtained by the GFIT3C fitting was integrated into the ABC code. To compare and validate the present case study, all calculations were performed by varying E from 0.4 eV to 1.4 eV at zero total angular momentum and diatomic parity +1.

The calculated $v = 0$, $j = 0$ state specific exchange probability is plotted in Fig. 3.5 as a function E. The agreement with the results of Ref. [151] is complete and provides a solid validation test of the two workflow approaches. For more detail, in Fig.. 3.6, a decomposition of the state specific probability into the state to vibrational ($v' = 0$ and $v' = 1$) ones is plotted using different lines and colors.

The test was extended by considering the

$$Li + HF(v, j) \rightarrow LF(v', j') + H. \tag{3.15}$$

exchange process having a more general nature. Not only, in fact, in process 3.15 the three involved bodies are different but also the interaction has a more structured shape that have been pinpointed in the past for leading to several peculiar dynamical features [152–166].

Such PES is slightly endoergic and shows a small barrier to reaction. Yet the endoergic nature of the reactive process turns into an exoergic one when including the corresponding zero point energy, allowing the exchange process to occur (in principle) with no help of collision energy [154, 155, 158].

[1]For the same benchmark calculation, we are experimenting the already mentioned alternative solutions searching on the web for suitable *ab initio* points or, if they are not available, starting the corresponding *ab initio* calculations.

ISSN: 2279-8773

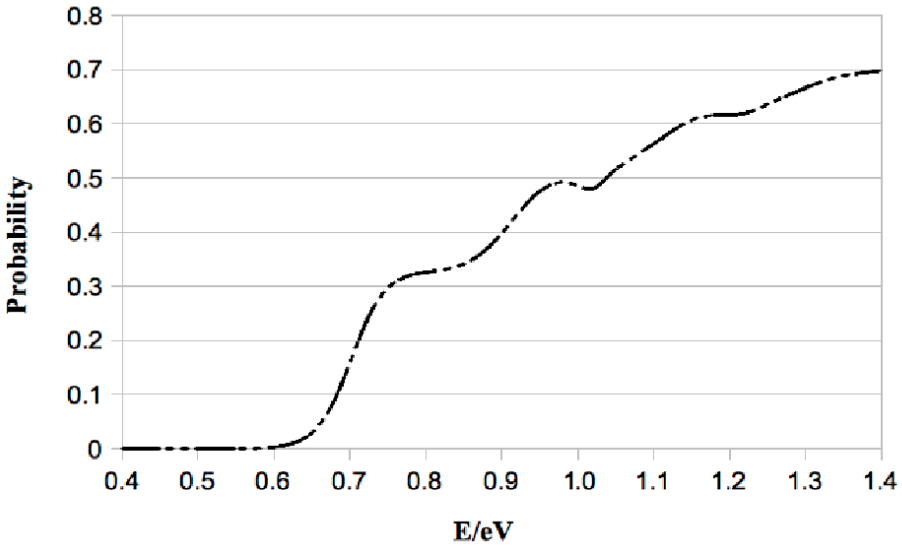

Figure 3.5: The exchange probabilities calculated at $v = j = 0$ and plotted as a function of E.

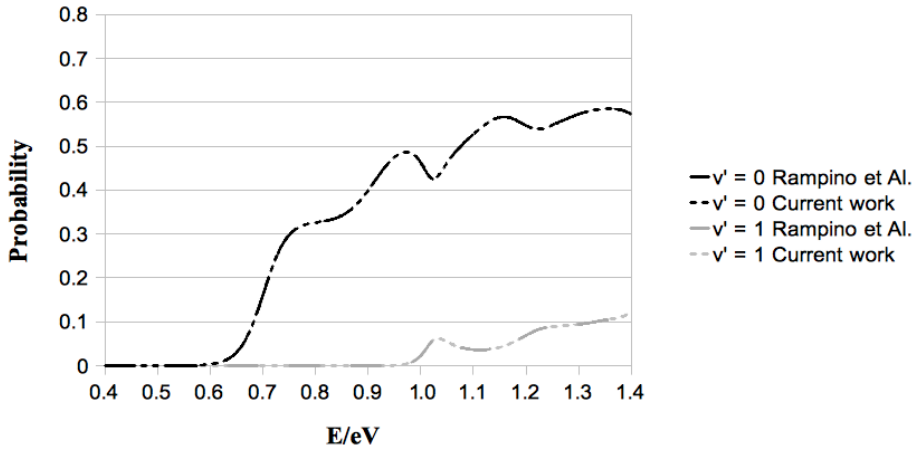

Figure 3.6: A detail of the exchange probabilities calculated at $v = j = 0$ for $v' = 0$ and $v' = 1$ plotted as a function of E.

ISSN: 2279-8773

Quantum calculations of the excitation function performed on the PLC PES reasonably well compare with the outcomes of previous computational campaigns as well as with related measurements. An important difference is singled out by the exploitation of the extra computational power offered by the Grid and that has allowed to study the reaction using a fine grid of energy values as shown in table 3.4. The key difference resides in the low energy portion of the excitation function that, in our case, increases significantly as energy lowers [154, 155].

To better assess the validity of the PES and in particular to check their accuracy in the reaction barrier region (that is typically the PES feature governing the low energy behaviour of the reactive cross section) of great importance is the recent experimental work performed at collision energy values ranging from 82 meV to 376 meV [167, 168]. More recently Loesch and coworkers [169] published data of variable angle crossed beam experiments in which collision energy was lowered down to 25 meV and provided clear evidence for a rising $\sigma(E_{\rm tr})$ as $E_{\rm tr}$ decreases below 0.1 eV (in qualitative agreement with the already mentioned calculations [154, 155] performed on PLC).

Using the GEMS workflow and exploiting the EGI Grid platform, the **S** matrix was calculated for an even finer grid of total energy values ranging from 0.25 eV to 0.45 eV (a steps of 0.0001 eV for energy values up to 0.27 eV and 0.001 eV in the higher energy value interval). The wavefunction was expanded into diatomic basis functions with internal energy smaller than 1.25 eV and propagated through 225 hyperradius sectors up to 24.0 bohr.

The J-shifting approximation [142] was used to extrapolate low J results to higher J values by assuming that the rigid rotor constant of the process transient $B = \hbar^2/2I_B = \hbar^2/2\mu_{\rm LiF}r^2$ is $3.59 \cdot 10^{-3}$ kcal/mol with $\mu_{\rm LiF}$ being the LiF reduced mass. Properties of the transient state of the Li + HF system were derived by calculating the minimum energy path at a value of the collision angle corresponding to that of the transition state. Transition state bond lengths were found to be 1.62, 1.31, and 1.73 $\mathring{A}$ for the LiF, HF, and LiH pairs, respectively.

The calculated $v = 0$, $j = 0$ probabilities (Ref. [170]) show that there is no collision energy threshold to reaction and that the reactive probability is constantly 1 in a small interval of near vanishing energies (for a range of 0.005 eV). Narrow resonance peaks are found in the low energy region (0.005 eV - 0.075 eV) while a smoother trend characterizes the higher energy region.

ISSN: 2279-8773

Table 3.2: Typical input parameters adopted for the present TI calculations.

Total angular momentum quantum number	0
Triatomic parity eigenvalue	+1
Diatomic parity eigenvalue	0
Maximum internal energy in any channel (eV)	1.25
Maximum rotational quantum number of any channel	200
Helicity truncation parameter	0
Maximum hyperradius (bohr)	24.0
Number of log derivative propagation sectors	150
Initial scattering energy (eV)	0.25
Scattering energy increment (eV)	0.001
Total number of scattering energies	10
Maximum value of v for which output is required	0
Maximum value of j for which output is required	2

The values of $\sigma(E_{\mathrm{tr}})$ calculated out of the obtained detailed reactive probabilities are plotted in Fig. 3.7. Contrary to previous results, they lead to a perfect agreement between theoretical and experimental quantities.

3.3.5 Performances and indications for further development

Application porting and the validation runs were initially performed on the SMSCG which runs on the Advanced Resource Connector (ARC) middleware [73]. The GC3Pie client script was run from a user interface machine at the Grid Computing Competence Center, University of Zurich (GC3).

GC3Pie allows to programmatically set the number of simultaneous jobs concurrently "in flight" (i.e., either running or submitted to a remote system and waiting to run); this is a feature used to better cope with the availability and responsiveness of the underlying infrastructure (no need to submit all possible jobs at once if the infrastructure cannot serve them).

For the validation test we set this number to 100 concurrent jobs. [2]

[2]This is an empirical value we chose mostly due to the limited number of sites support-

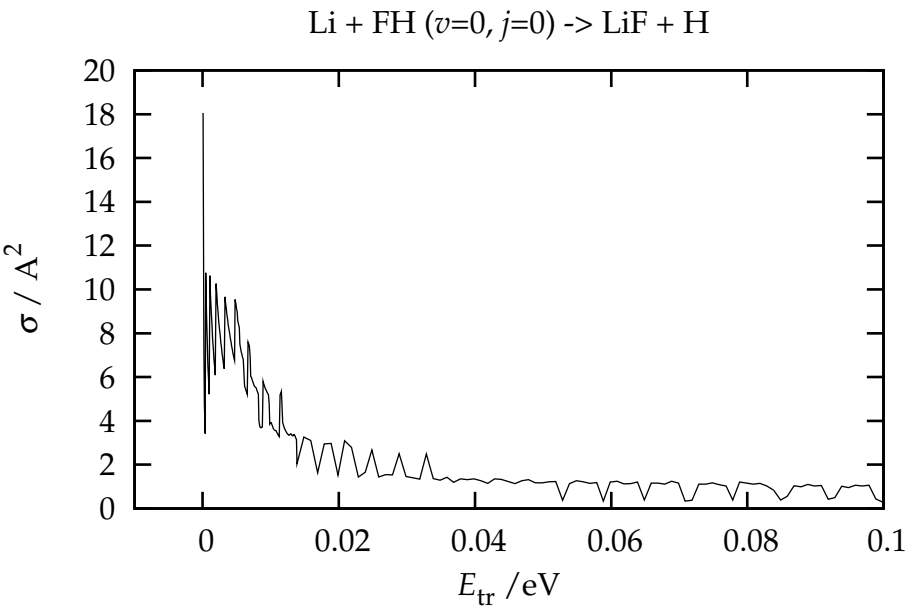

Figure 3.7: Plot of the J-shifting estimate of the cross section as a function of translation energy.

After about 32 hours, we were able to collect the results coming from 712 successful job submissions, for a grand total of about 138000 min of computation time, i.e., about 2300 CPU-hours. The remaining jobs failed due to hardware and/or software related problems that need further investigations. The average run time of successful jobs was thus of about 190 min, or little more than 3 CPU-hours. Fig. 3.8 provides a pictorial representation (carried out by using the code provided by Panse [171]) of the running and waiting time for each individual job.

The gaps in the visualization should be interpreted as a concentration of failed jobs in a short time span; a cross-analysis of the job error messages shows that two clusters of SMSCG had a malfunction and acted as "black holes".

In Fig. 3.9 the number of running and pending jobs as a function of time in the current grid submission are shown. A simulation of an ideal submission of 100 jobs constantly running and no queue wait time (see Fig. 3.10) takes about 26 hours to complete. According to our simulations, the grid execution model added less than 25% overhead for a realistic use case, involving significant waiting time at remote clusters

ing the AppPot application tag at the time we run the whole validation test.

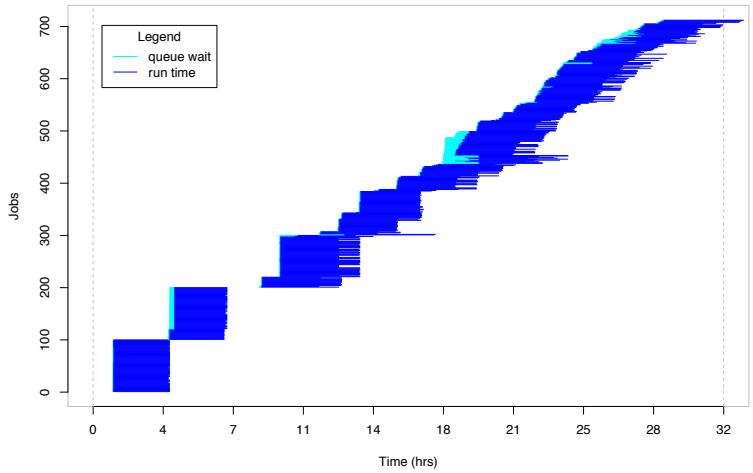

Figure 3.8: Representation of running and waiting times for the successful jobs in the SMSCG workflow submission experiment. Each horizontal segment represents a job; the light- and dark- colored portions of the plot measure the waiting time (in the remote batch system queue) and the actual running time, respectively. The gaps should be interpreted as a concentration of failures in a certain time span.

and important malfunctions.

As a result of the work reported above we can conclude that the porting of legacy computational chemistry applications onto the Grid infrastructure, together with the development of the related workflows, has led to the building of a solid platform for assembling accurate multiscale realistic simulations and for establishing an advanced molecular and material science research environment.

In particular the library of quantum mechanics set of codes of GEMS analyzed in the first part of this section and devoted to few body calculations assembled to be offered offered to the users of the COMPCHEM Virtual Organization for production has shown to be a useful development field for workflows and performing implementations on Grid. On this ground the adoption of the framework GC3Pie allowed us to define event-related dependencies between different applications and execute them simultaneously on a large-scale distributed computing infrastructure. The main difference with other popular workflow systems is the programmatic approach to workflows. In GC3Pie there is no fixed and pre-defined structure of the workflow: the entire execution schema is

ISSN: 2279-8773

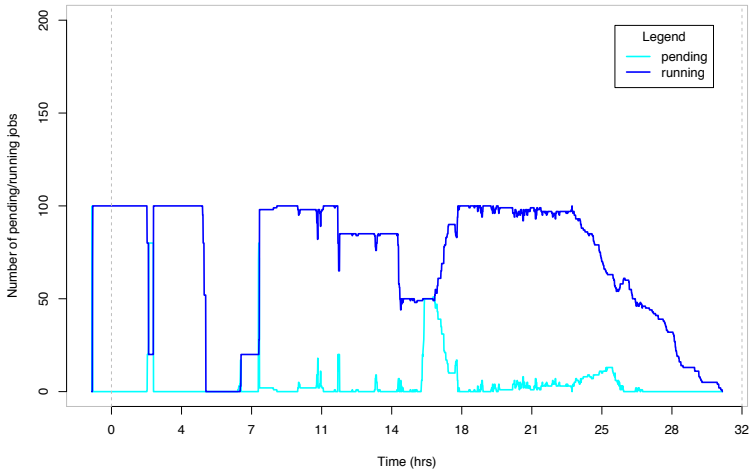

Figure 3.9: Number of running (dark color) and pending (light color) jobs plotted as a function of time in the SMSCG infrastructure submission. The GC3Pie client was configured to check job status every 45 seconds, and try to keep a total of 100 jobs in submitted or running status at any time.

assembled at runtime and steps can be added and removed dynamically as the program progresses, adapting to the outcome of individual computations. The implemented case study demonstrates the validity of this approach by following different routes depending on the availability of outcomes of previous investigations. Such success paved the way to further work on extending workflows to the molecular dynamics of multi-body systems.

3.4 Workflow extensions to Multi-body systems

The simulation of complex multi body systems starting from first principles is nowadays becoming increasingly popular thanks to the possibility of adopting computational approaches splittable in independent computational tasks. This property makes related computer codes the tools of election for carrying out massive calculations and represents a strong incentive to implement them on Grid platforms by developing appropriate distribution models.

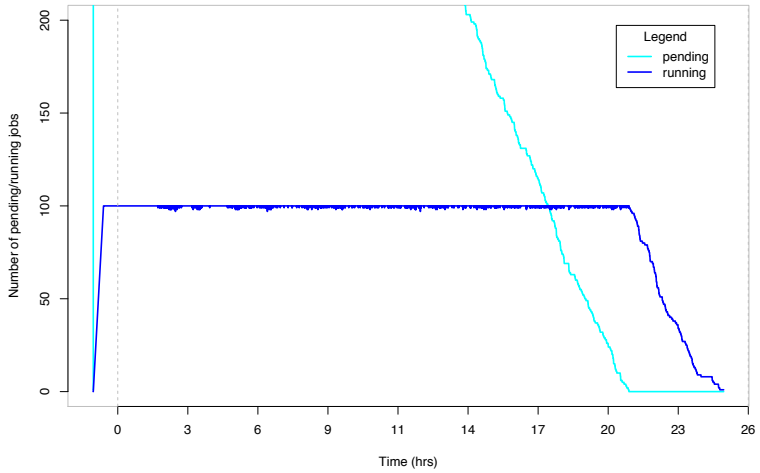

Figure 3.10: Number of running and pending jobs plotted as a function of time in a simulated "ideal" situation. In this simulation, all jobs are submitted at the onset, and then are executed on a dedicated cluster of 100 independent CPUs; it is further assumed that no other jobs are contending the use of the cluster. The solid blue line clearly shows that the number of running jobs is roughly constant until the end, when there are not enough pending jobs to fill the cluster.

For this purpose a collaborative work has been carried out to port within GEMS on the EGI Grid environment some Molecular Dynamics (MD) packages like GROMACS [10]. This has allowed some members of the COMPCHEM VO to intensively use molecular simulation tools and develop appropriate visualization tools facilitating the inspection and rationalization of the molecular dynamics outcomes.

To this end the P-GRADE portal [172] has been used. P-GRADE is an open source Grid Portal [172] that provides intuitive graphical interfaces to facilitate the porting and does not necessarily require the modification of the original code for its distributed execution on the Grid.

The complexity of the MD applications and the flexibility of P-GRADE has motivated us to develop a new distribution strategy in which the execution is distributed using different computing environments: while the main tasks are executed on the Grid, some tasks are executed on a local cluster and other tasks require the access to a proprietary database.

ISSN: 2279-8773

Here, the new alternative strategy of using external services (SE) and proprietary applications in COMPCHEM is discussed by exploiting the potentialities of a new workflow developed at CESGA (see Ref. [173]). These enhancements enable the final user to perform complex simulations by combining different softwares and using the output obtained by proprietary applications or stored in private databases as input for CPU demanding applications running on the Grid. A preliminary analysis on how to extend the work performed for quantum mechanics treatments of few body problem to MD simulations has been first carried out for GROMACS and then generalized to other MD packages.

3.4.1 The GROMACS program overview

GROMACS is a versatile package performing MD calculations for hundreds to millions of particles. GROMACS is primarily designed to handle large biochemical molecules like proteins and lipids, which exibit quite complex patterns of bonded interactions. GROMACS is also used by several research groups to handle large non-biological systems (like polymers) because of its speed in calculating also non-bonded interactions (which usually dominate molecular simulations).

GROMACS performs well on scalar machines and scales up satisfactorily with the number of processors on parallel machines. As a matter of fact, GROMACS showed to encompass a minimal-communication domain decomposition algorithm, full dynamic load balancing, a state-of-the-art parallel constraint solver, and efficient virtual site algorithms, which allow the removal of the hydrogen atom degrees of freedom allowing the use of integration time steps as high as 5 *fs* for atomistic simulations. To improve the scaling properties of the common particle mesh Ewald electrostatics algorithms, a Multiple-Program, Multiple-Data approach has been used with separate node domains responsible for direct and reciprocal space interactions [10]. This combination of algorithms enables extremely long simulations of large systems and leads to similar simulation performances on modest numbers of standard cluster nodes.

3.4.2 The workflow articulation of GROMACS

The distributed version of GROMACS has been implemented on the Grid so as to execute its scalar version on multiple Grid resources with

ISSN: 2279–8773

different input files simultaneously (the so-called "parameter study" modality [174]). These simulations can be distributed on a large number of computers because they can run independently from each other. As Grid resources are more exposed to faults than supercomputers, the execution was made fault-tolerant by using a Grid-specific component able to manage the execution of these jobs, collect Grid resources, transfer the code with the corresponding input files to the computing elements, start the jobs, observe and supervise their execution and finally even stage out the result files after successful completion.

As we have already mentioned the extension of the gridification work to MD making use of P-GRADE. P-GRADE, in fact, is a Grid Portal that provides graphical tools and services suited to help Grid application developers in porting legacy programs onto the Grid without reengineering or modifying them. P-GRADE has the structure of a workflow enabling application developers to define the above-mentioned parameter study by means of a graphical environment and to generate out of the user-specified description the Grid scripts and commands allowing the execution of the various computational tasks on the distributed Grid platform.

P-GRADE integrates batch programs into a directed acyclic graph by connecting them together with file channels (a batch program can be any executable code that is binary compatible with the underlying Grid resources). In our case, the Grid resources are typically the gLite middleware [74] and a subset of the EGI processors. The file channel defines directed data flows between two batch components like the output file of the source program used as input file of the target program. The workflow manager subsystem of P-GRADE resolves such dependence during the execution of the workflow by transferring and renaming files. After the user has defined the structure of the workflow and has provided executable components for the workflow nodes, it has to be described, whether to execute the workflow with just one input data set, or with multiple input data sets in a parameter study fashion. If the latter option is chosen then the workflow manager system of P-GRADE creates multiple instances (one for each input data set) and executes the workflow instances simultaneously.

The Portal server acts as a central component to instantiate workflows, manage their execution and perform the file staging, which involves input and output processes. Moreover, the P-GRADE also provides tools to generate input data sets for parameter study workflows automatically. The user has the possibility of attaching the so-called "Genera-

ISSN: 2279–8773

tor" components to the workflow in order to create the parameter input files. The workflow manager subsystem executes first the Generator(s) and then the remainder of the workflow, ensuring that each workflow accesses and processes the correct and different permutations of the data files.

A simple workflow: the water case study

In order to test the workflow developed for GROMACS we used as a case study the "Water: Energy minimization of the solvated system" described on the GROMACS tutorials [175]. In this case study the various jobs work with different "Temperature" values for "LANGEVIN DYNAMICS OPTIONS" in a typical scaling temperature technique. Accordingly, the values of the `bd-temp` (see Table 3.3) temperatures considered are stored in the same input file as the other input parameters of the simulation. Before the central component can start running the GROMACS jobs, other components need to generate all the necessary input files to be used as input during file stageing to Grid resources.

For this reason the central component of the workflow (shown as a box labelled GROMACS SEQ in the left hand side panel of Fig. 3.11) is made of a bash script in which the following steps are performed:

- download of the GROMACS executable from a COMPCHEM server;
- configure the enviroment variables;
- execute the GROMACS executable (already compiled on the User Interface machine of the EGI Grid).

The smaller boxes attached to this central component represent the input and output files which are used and produced by the application. During Grid execution the P-GRADE workflow manager is responsible for preparing these files for the Fortran program and transferring them to the EGI Computing Element. This makes the executable know nothing about the Grid and no modification is required.

The first workflow component is an Automatic Generator (see the GROMACS A_GEN box in the upper region of Fig. 3.11). The automatic Generator is a special job type in P-GRADE. It is used to generate input text file variations for the GROMACS job. Using the

ISSN: 2279–8773

parameter definition window of the Automatic Generator the Temperature `bd-temp` values have been defined for the bench concurrent simulation (see the right hand side panel of Fig. 3.11). As the parameter definition window of P-GRADE is highly flexible, we could reuse a generic input file of GROMACS and put a parameter key (say p_1 as in the figure) into it. This parameter key is automatically replaced by the actual parameter values during the execution of the workflow to generate the input files before the GROMACS simulation jobs are started.

The third component of the workflow is a Collector job (see the GROMACS COLL box in the lower region of Fig. 3.11) which is again a special job type in P-GRADE. A Collector in a P-GRADE workflow is responsible for collecting the results of the parameter study workflow, analyzing them and creating a typical user friendly filtered result (to be considered as the final result of the simulation). In our case the Collector simply collects the results from the GROMACS jobs and compresses the files into a single archive file that can be downloaded by the user through the Portal web interface. The purpose of this step is, in fact, to make the results of the computations accessible by the end users.

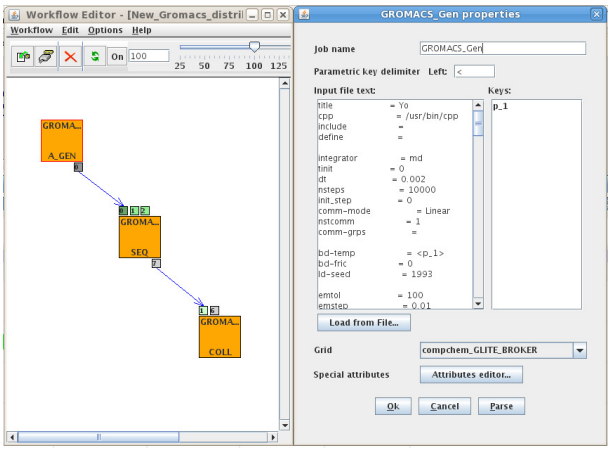

Figure 3.11: A schetch of the workflow developed for GROMACS package.

Table 3.3: Main input parameters used by GROMACS in the benchmark calculation.

Parameter	**Comments**
; RUN CONTROL PARAMETERS	
`integrator` = md	Integrator
`tinit` = 0	Start time in ps
`dt` = 0.002	Timestep in ps
`nsteps` = 10000	Number of steps
`init_step` = 0	Initial step
`comm-mode` = Linear	Mode for center of mass motion removal
`nstcomm` = 1	Number of steps for center of mass motion removal
; LANGEVIN DYNAMICS OPTIONS	
`bd-temp` = 300	Temperature
`bd-fric` = 0	Friction coefficient (amu/ps)
`ld-seed` = 1993	Random seed
...	

3.4.3 Performances measured in COMPCHEM

The gridified version of GROMACS was made execute the input generator and the GROMACS code simultaneously on multiple Grid resources. GROMACS was compiled in a static way using the standard gnu compiler gcc/g77 and the open source atlas library. Static compilations of GROMACS ensure that the program is binary compatible with the used Computing Elements of the EGI Grid and that the program will not run into incompatibility errors associated with the usage of dinamically loaded libraries. As we did not modify the code any performance improvement can be ascribed only to the Grid implementation that means to the possibility of running the package for different sets of parameters during the same time window on the resources of the COMPCHEM VO.

COMPCHEM relies on more than 8000 CPUs located at more than 25 European research institutes and universities with most of them

ISSN: 2279-8773

being shared among several VOs. As a consequence jobs sent by COMPCHEM members to these Grid resources compete for CPUs with both other jobs of COMPCHEM and jobs of other VOs. For example, a run of 4 GROMACS jobs in a parameter study fashion for the already mentioned benchmark case took the average time of 100 minutes when using 3 CPUs while the average time taken on 6, 12 and 24 CPUs was 91, 113 and 171 minutes respectively. This allow us to extrapolate an average single CPU time of 33, 15, 9 and 7 minutes respectively. This means that the distribution of GROMACS on the Grid leads to a definite reduction of the average single CPU time if the parameter space is larger than 3.

As the execution time of both the generator and the collector stages are neglegible compared with that of the executor, we can assume that the figures of time consumption quoted above roughly coincide with those of the application as such.

This means that a GROMACS job can spend on the average as much time in the job queue of a single CPU of a Grid resource as on the CPU itself. This obviously means that the average execution time of a job on the Grid is about twice as long as that on a dedicated local machine or that the Grid execution adds about 2-8 hours to each job. Accordingly, as soon as there are at least more than 3 GROMACS jobs running concurrently in a simulation, the Grid based execution is more efficient.

3.4.4 A new distribution schema: merging local Clusters and Grid resources

First workflow approach

A new distribution strategy inspired to the one developed by CESGA for the G-Fluxo Project [176]) has been applied to the GROMACS package. As a result a first workflow have been implemented and tested using the P-GRADE Grid Portal for one of the available GROMACS tutorials [175] to exploit the interoperability between a local cluster platform (HPC capable) and a Grid platform (mainly HTC capable). The three jobs considered for this purpose are:

1. Vacuum: Energy minimization of the structure.
2. Water: Energy minimization of the solvated system.

ISSN: 2279-8773

3. PR: Relaxation of solvent and hydrogen atom positions: Position restrained Molecular Dynamics.

In this case the workflow is executed in the FinisTerrae cluster (1st job), the SVG cluster (2nd job) and the EGI Grid (3rd job). The coupling among the various jobs is taken care using a link (a semaphore) that defines the dependency job chain. In this case the user must set all the job ports adequately following the syntax described previously in Ref. [176], taking into account where the job is executed so in order to make the portal aware of all the information needed for file transfers. This is needed to overcome the limitation present in the P-GRADE Grid Portal version 2.7 file transfer management system that does not allow direct file transfer between different platforms (different Grid middlewares following P-GRADE Grid Portal nomenclature).

Second workflow approach

The second workflow, also implemented and tested for GROMACS, is made up of a chain of three jobs where each job runs on different platforms (local cluster, CESGA SVG, a laptop and the Grid infrastructure) [173]. This workflow takes advantage of some extra functionalities added to the P-GRADE Grid Portal and in particular:

- File Management and Communication via SSH protocol between different platforms taking into account the dependencies coming from the workflow definition;
- Job Submission and Monitoring using DRMAA implementation included in the Distributed Resource Management Systems (DRMS) in order to use local resources;
- External services called via web making use of the POST/GET html protocol.

It makes use of the following common tools and applications to perform accurate simulations. A schematic representation of the workflow assembled for the study reported in the present paper is illustrated:

1. CSD (Cambridge Structural Database [177]): the CSD is accessed through a CESGA server (SVG). This job runs ConQuest in batch mode through the command "cqbatch". A simple query is performed looking for compounds whose name includes the term

ISSN: 2279-8773

"azaindole" included. As a result a compressed file (azaindole.tgz) in which all the structural 3D information in form of PDB files coming from the previous query are stored, is returned.

2. PRODRG: This job takes as input the azaindole.tgz file and runs a curl [178] script to make use of an external web server, the Dundee PRODRG2 Server [179] that provides molecular topologies for X-ray refinement/MD and drug design/docking. This job runs from the laptop connected to internet and configured as a CLUSTER-GRID accessible by ssh. As a result a compressed file called azaindole-prodrg.tar, containing the GROMACS ITP files [180] for all the compounds coming from the CSD query, is created.
3. GROMACS: This job runs an energy minimization for every structure made of a box of 1000 water molecules using as input the files contained in the azaindole-prodrg.tar file. It can be easily tuned to run a minimization of the compound with a protein structure (as it is usually performed in protein-ligand complexes studies) relevant to drug design. All the results are stored in a compressed file directly downloadable from the portal.

Workflow details

The workflows used are shown in Fig. 3.12 and 3.13 respectively, together with their definition. Relationships among the various jobs are expressed using a link (this involves a file transfer in each case) that defines the dependency job chain. In this case the user must set all the job ports by following the syntax described in Ref. [173] and taking into account where the job is executed so that the portal could be aware of all the information needed for file transfers. In the case of the GROMACS job (to be executed on EGI) the file should pass through the portal using the local file feature. This is needed to overcome the limitation present in the P-GRADE File Transfer Management System that does not allow direct file transfer between different platforms (different Grid middlewares following the P-GRADE Grid Portal nomenclature). A modification of P-GRADE could enable the direct file transfer between a CLUSTER-GRID and different Grid middleware.

Another interesting improvement would be the connection of a sequential job to an Automatic Generator directly. At present this step needs to be performed separately by the user. Modifications in P-GRADE devoted to overcome this limitation are under study.

ISSN: 2279–8773

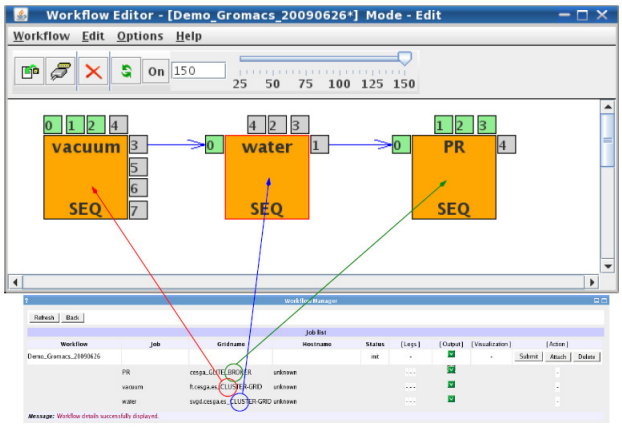

Figure 3.12: First workflow Schema as it is presented by the P-GRADE Grid Portal workflow editor. The workflow involves a coordinated execution using external services, Cluster platforms and Grid infrastructure.

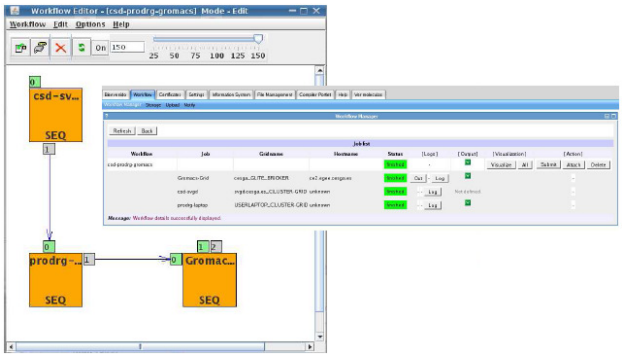

Figure 3.13: Second workflow Schema as it is presented by the P-GRADE Grid Portal workflow editor. The workflow involves a coordinated execution using the Grid, a Cluster platform and a user computer.

Computing platform and workflow support

As the portion of the code in P-GRADE devoted to send jobs to the Grid environment has not been altered, a significant effort has been devoted to develop a set of scripts (fully integrated into the Portal) able to manage the submission of jobs to local cluster. To do that the DRMAA standard library for the job submission and monitoring to Distributed Resource Management Systems (DRMS) has been used and the scripts have been compiled and tested on the two Sun Grid

ISSN: 2279-8773

Engine (SGE) platforms present at CESGA: SVGD and Finis Terrae.

The addition of a local cluster in P-GRADE is obtained by defining in the configuration files of the Portal a new type of Grid and copy it in the account of each user present in the local cluster the generated SSH public key as shown in Fig. 3.14. Following this procedure the remote accounts are exposed to the portal, but not to the other users having an account on the portal.

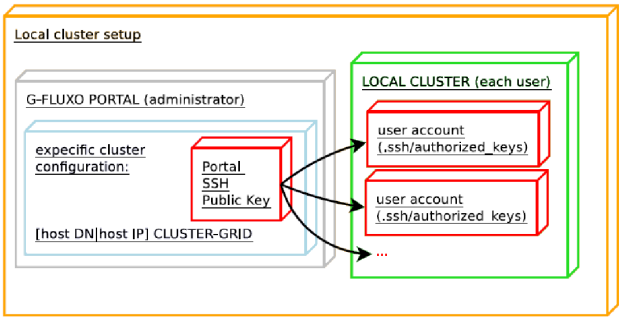

Figure 3.14: SSH configuration for Local Cluster

To copy files between local cluster and grid as well as local cluster and server, a new syntax has been specifically defined into the workflow editor:

cluster:[host DN—host IP]:[path to file]

and for each job a special folder is created in the local cluster where the files needed for the job execution are copied. In order to implement such skills on P-GRADE, three scripts have been developed wkf_ pre_ CLUSTER-GRID.sh, wkf_ post_ CLUSTER-GRID.sh and ff_CLUSTER-GRID.sh. To manage the input files needed for the calculations, another script (wkf_pre.sh) has been developed and integrated into the Portal. The script calls specific functions on the local cluster located in the file wkf_ pre_ CLUSTER-GRID.sh. At the same time to manage the output files carried out from the calculations, the wkf_post.sh script has been developed and integrated into the Portal. The script calls specific functions on the local cluster located in the file wkf_ post_ CLUSTER-GRID.sh. To perform the file management on different resources and technologies (such gLite, for dealing with the Grid platform, and SSH, for local cluster) the ff_CLUSTER-GRID.sh

has been developed. Each user can request the needed amount of resources belonging to the local cluster for the job execution. Those resources are specified by variables and parsed through an ad-hoc script. However, since the workflow editor used by P-GRADE is unable to link different resources, all files need to transit trough the P-Grade Portal Server (in the future the workflow editor will be modified to overcome this limitation).

As an example, the architecture diagram showing the pre/post file management and job submission procedures is presented in Fig. 3.15. In the same Figure the communication protocols SSH based (SCP/SSHFS) linked with the specific tasks related with the different functions developed in the ff_CLUSTER-GRID.sh script are also shown.

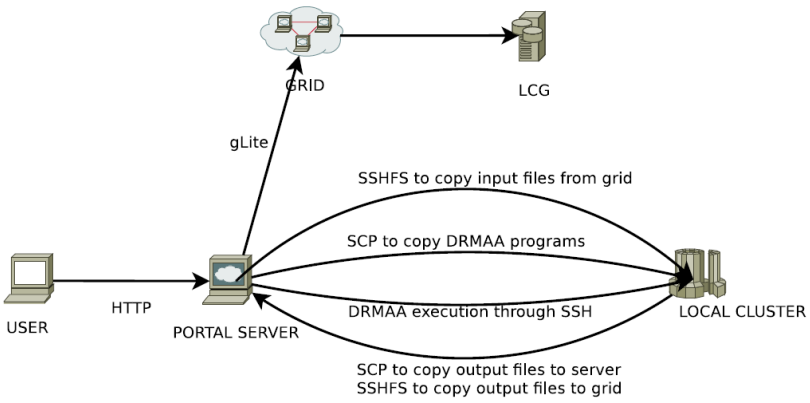

Figure 3.15: Computational platform architecture mixing HPC and HTC resources

3.4.5 Added value: a specialized Visualization Tool

As an added value a set of visualization tools has been developed as portlets and implemented in the P-GRAGE Grid Portal to be deployed in GEMS.

The use of portlets allows to easily add new tabs and windows associated with simulation and/or grid file management applications needed by the user. As P-GRADE is entirely developed using Java, JSP and GridSphere, these three components were used to develop a specific portlet for the visualization of the GROMACS output.

ISSN: 2279-8773

For this purpose two code files (written using JSP and Java, respectively [18]) and the corresponding configuration files have been developed together with a Jakarta Ant script to deploy the portlet in P-GRADE. At the same time the Jmol package [181] was used as a specialized visualization tool. Jmol is an open-source Java viewer for chemical structures which contains a web browser applet (JmolApplet) that can be integrated into web pages or, as in our case, into a GridSphere Portlet.

The main class in Jmol Portlet is 'UiJmol'. That class is a child of 'ActionPortlet' class of GridSphere. The 'UiJmol' class calls 'uiJmol.jsp', in which the main layout of the web page is described. The portlet layout consists of a form devoted to choosing the file to visualize, and of an applet devoted to rendering the molecule. The form was developed using three boxlists (`ListBoxBean` class). The first boxlist is used to choose the workflow, the second to choose the job in the workflow, and the third to choose the molecule file (output of GROMACS). The portlet searches for the files only in the current user account of P-GRADE (since it can not access files in other user's accounts). In the visualization process the HTML code (as wellas the JavaScript code) is used to call the Jmol applet specifying the location of the output file for the visualization.

Jmol supports several input file formats, but does not support those of GROMACS. As a solution for that problem, the `pdb2gmx` command, integrated in the GROMACS package, is used to convert the GROMACS output to Protein Data Bank (PDB) file format [182]. All the described components are packed together in a single tar file for an easy distribution and the deployment can be done with an ad-hoc Jacarta Ant script. In Fig. 3.16 a screenshot coming from the visualization Jmol portlet implemented in COMPCHEM P-GRADE Grid Portal is shown. Using this portlet the PDB file obtained from the calculations can be easily visualized. All the functionalities available at the JmolApplet are also present in the portlet. Even more, Jmol additional functionalities like the RasMol/Chime scripting language and JavaScript support library can be integrated into a very specific visualization portlet and there is no need to wait for completion of the whole workflow.

ISSN: 2279–8773

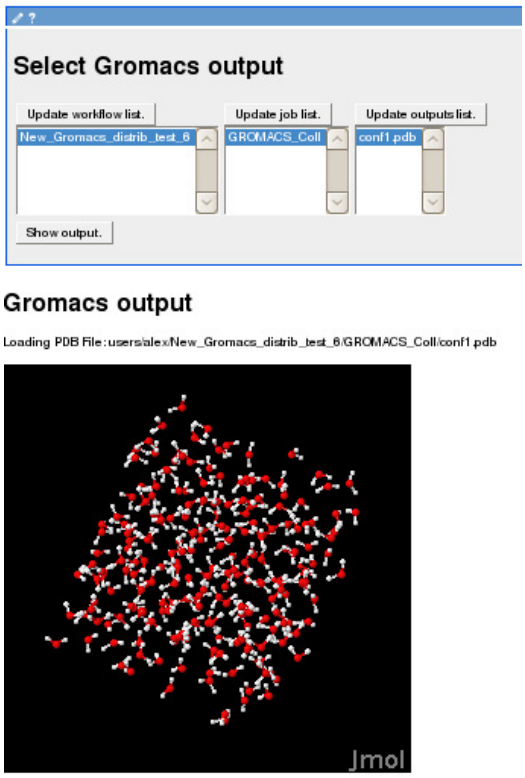

Figure 3.16: Sketch of the Output visualization for the developed multiplatform GROMACS workflow.

3.4.6 The DL_POLY program

The program overview

A similar investigation was started for the DL_POLY [11] because of its popularity and of its extended usage by the Computational Dynamics and Kinetics group of University of Perugia. DL_POLY is a package of subroutines, programs and data, designed to facilitate MD simulations of macromolecules, polymers, ionic systems, solutions and other molecular systems on a distributed memory parallel computer. The package was written as part of the UK project CCP5 by Bill Smith and Tim Forester on a grant of the Engineering and Physical Sciences Research Council and is the property of the Science and Technology Facilities Council (STFC).

At the time of the investigation, two versions of DL_POLY were avail-

able. DL_POLY_2, the earlier version, based on a replicated data parallelism. It is suitable for simulations of up to 30,000 atoms on up to 100 processors. DL_POLY_3 (written by I.T. Todorov and W. Smith) instead, based on domain decomposition parallel schema and designed for systems beyond the range of DL_POLY_2 - up to 10,000,000 atoms (and beyond) and 1000 processors.

The Grid implementation of DL_POLY

To investigate how Grid-targeted modifications of the distribution strategies would work for DL_POLY, it was implemented on the EGI production Grid platform available in COMPCHEM.

As a first step, to measure the performance of the code on the Grid and to single out the Grid features exploitable for the purpose of improving the statistics of the selected events, we ran the parallel version of the DL_POLY suite of codes, based on the Replicated Data parallelization strategy [183] on six different EGI-Grid clusters of processors. In order to evaluate the elapsed time of each simulation and the related speed-up for each cluster, we ran the calculations sequentially on one node and in parallel on 2 and 4 nodes.

As a second step, the parallelization of DL_POLY has been carried out at coarse granularity, by adopting a task farm model. In this simple model, a master process distributes the whole program to all the available Grid nodes and executes it in parallel for different values of the parameters, in a typical SPMD (Single Program Multiple Data) fashion. In particular, we distributed different calculations for various temperatures. This model has the advantage of being simple to implement and to be, in some way, the one most suited for the present structure of the EGI Grid.

Related calculated elapsed times and speedups measured on different EGI Grid clusters are plotted in Fig. 3.17 and 3.18 respectively. As shown by the figures some clusters on the EGI-Grid have a parallel performance very close to the ideal value because they allow a dedicated usage of the processors. Deviations from the ideal value do not necessarily depend only on the time sharing regime adopted by some clusters. To investigate this aspect a detailed evaluation of the parallel performances of the clusters and of the waiting time intercurring between the scheduling and the running of a process were obtained by restricting parallel calculations to two nodes. The results obtained by

ISSN: 2279–8773

running 50 parallel jobs showed that more than 70% of the jobs ran properly and only 26% were aborted. The main reason for abortion was found in communication errors between the nodes of the same cluster (62%). About 15% of the failures were due to faults of the scheduler and another 23% to internal errors of DL_POLY at run time.

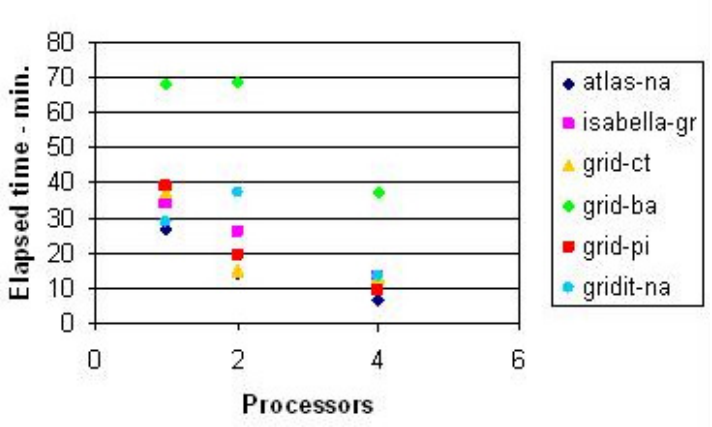

Figure 3.17: Elapsed time measured on different EGI-Grid clusters plotted as a function of the number of processors used (see online version for colours)

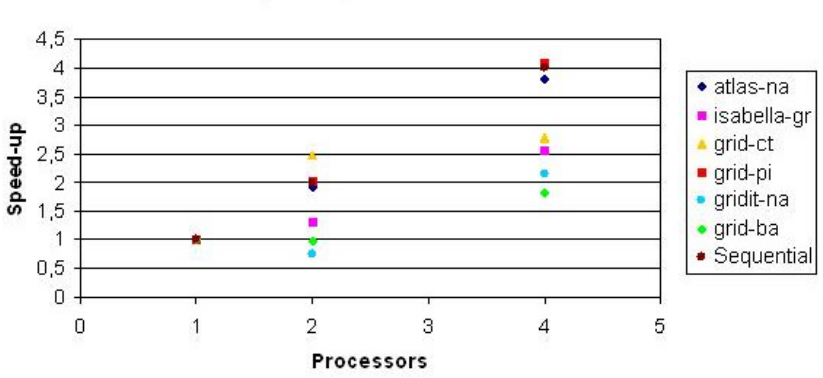

Figure 3.18: Speed-ups measured on different EGI-Grid clusters plotted as a function of the number of processors used (see online version for colours)

Indications for an alternative distribution scheme

The non-negligible failure rate of the calculations due to internal DL_POLY problems (and not to networking problems) prompted the need for de-

ISSN: 2279-8773

veloping alternative distribution strategies. This has motivated specific work on the DL_POLY package itself.

In fact, whereas failures due to either communication or scheduling reasons can be remedied by re-running the related computation, while waiting for the Grid to become more fault-tolerant, internal DL_POLY failures imply that one or more starting conditions are inappropriate and make it difficult to properly sample initial conditions. To deal with this problem we examine in the followings the algorithms used for increasing the statistical significance of the sample after the thermalization process.
As is well known, the average value $< A >$ of property A of a given system of N particles is defined as

$$< A >= \int \int dp^N dr^N A(p^N, r^N)\rho(p^N, r^N) \tag{3.16}$$

where $A(p^N, r^N)$ is the property A expressed as a function of the phase space variables p^N and r^N while $\rho(p^N, r^N)$ is the probability density of the ensemble. Usually, as already mentioned, in Molecular Dynamics simulations $< A >$ is approximated in the ergodic approach by the time average $< A >_{time}$ defined as

$$< A >_{time}= \lim_{t\to\infty} \frac{1}{t} \int A(p^N(t), r^N(t))dt \simeq N_\tau^{-1} \sum_{t=1}^{N} A(p^N(t), r^N(t))$$

where N_τ is the number of sampling points considered. Accordingly, after partitioning the integral in N_k subsets we can also write

$$< A >= N_k^{-1} \sum_{k=1}^{N_k} < A >_k= N_k^{-1} \sum_{k=1}^{N_k} N_{\tau k}^{-1} \sum_{t=1}^{N_{\tau k}} A(p^N(t), r^N(t))$$

This leaves us with the choice of selecting appropriate configurations of the system that can be obtained by randomly altering the after thermalization configurations having the same energy. In our approach we randomly invert the sign of a fixed number of momentum components of the particles of the system without changing their modulus. As a result, the portion of the phase space sampled by the system changes slightly without altering the global and individual total energy of the particles.

To implement our selection algorithm, new MPI constructs have been added to the original code. Use has also been made of the preprocessor

ISSN: 2279-8773

instructions to avoid the compilation of the original MPI implementation of the DL_POLY package as described in [184] and modifications have been introduced in the workflow, merging the distribution strategies previous described in section 3 and 4. Further extended tests are being carried out to better evaluate the performances of the proposed approach.

3.5 Moving to a High Performance Grid

The implementation of real systems on the Grid require often to go beyond the present structure of the Grid platform. As a case study we considered the rationalization of gas hydrates formation. Gas hydrates are ice-like solid inclusion compounds which result from the trapping of gas molecules within a lattice-like cage of water molecules. Many gases have molecular sizes suited to form hydrates like methane (CH_4) and carbon dioxide (CO_2). Gas hydrates are essentially water clathrates in which the water cage cristallizes in the isometric cristallografic system rather than in the hexagonal one of normal ice [185]. Actually, the cage of water molecules (*host*) is stabilized by the trapped gas molecule (*guest*) and, without such contribution, the lattice structure of hydrate clathrates would collapse into a conventional ice crystal structure or liquid water.

Clathrates became famous some time ago because of their ability to obstruct natural gas pipelines [186]. Light gases, such as methane or ethane, always present in oil products can, in fact, at low temperature and high pressure, get trapped as guest molecules in hydrate structures and make them solidify [187]. However, important benefits can be derived from the exploitation of this property. For instance, at standard conditions of pressure and temperature, one volume of methane hydrate can store up to 164 volumes of gaseous methane [187], allowing its safer and less expensive storage and transportation. Because of this, methane hydrates can be considered the most abundant carbon-based energy source on Earth [188, 189] (conservatively estimated to amount to about twice the quantity of carbon in all known fossil fuels). this makes the sedimentary methane hydrate reservoirs a potentially important source of hydrocarbon fuels. Moreover, the lower stability of the CH_4 clathrate with respect to that of CO_2 has suggested the use of extraction of methane from clathrates together with the adoption of carbon dioxide mitigation techniques based on the sequestering of CO_2

ISSN: 2279–8773

in hydrates [190, 191].

The mechanisms of formation of a gas hydrate involves typically the following steps: gas dissolution, formation of crystallographic nuclei and their growth. The bottle neck for such mechanism is the formation of the solvent cage that needs long induction time and has a low rate [192]. On the theoretical side, the determination of the steps of such mechanism and of the structure of the water cages formed by the host molecules (which involve non covalent forces [190, 193]) prompts an accurate description of the interaction and a rationalization of the aggregation forms of the solvent in the presence of the guest gas. On its side the experiment has already indicated that the formation of the solvent cage depends on the interfacial area, pressure, temperature as well as on the extent of supercooling [194]. The experiment has also singled out that, after a certain induction time, the structure of gas hydrates grows mainly at the water-gas interface [188] with the rate of formation of the clathrate being controlled by the gas diffusion rate through the hydrate film.

The gas diffusion is enhanced by some additives, like the surfactants (*surf*ace *act*ive *a*g*ents*), whose molecules have lipophilic and hydrophilic moieties (amphiphilic molecules). This property has been rationalized in terms of a reduction of the surface tension at gas-water interface by the surfactant that facilitates the gas diffusion into water [195]. To this end, Zhang et al. [196], when investigating the behaviour of sodium dodecyl sulfate (SDS) solutions, suggested that SDS molecules adsorb on hydrate nuclei and reduce the energy barrier for further aggregation. This was rationalized, at least in a certain concentration range, in terms of a tendency of the surfactant molecules to form micelles which can act as effective sites for hydrate nucleation [197]. Yet, a few years later, thanks to the analysis of the Critical Micelle Concentration (CMC) of several surfactants, it was suggested [198] that no micelles intervene during the formation of the gas hydrate. To put that finding on a robust theoretical ground we started extended calculations aimed at modeling the mechanism of formation of gas hydrates when surfactants are added to the water-gas system [199–202]. To this end, a recently developed model potential (see for instance Refs. [203–210]), based on the decomposition of the molecular polarizability into effective components and using a new formulation of non-covalent interactions, has been utilized to build an appropriate Potential Energy Surface (PES). Such PES has been used to carry out Molecular Dynamics (MD) simulations for systems containing SDS-CH_4, SDS-H_2O and CH_4-H_2O pairs

ISSN: 2279-8773

by running the DL_POLY [11] package. The package was implemented making use of the cluster of the Departamento de Quimica Fisica at the University of Barcelona and on the EGI Grid Infrastructure available to the virtual organization (VO) COMPCHEM [6].

The present study concerns the investigation of a system made of one molecule of SDS and one molecule of CH_4 in an environment of 2000 water molecules. The study showed that, even in extremely diluted cases in which the formation of micelles is impossible, SDS is able to organize water molecules in a way that the methane molecule is driven (and then dropped) inside their cage. In spite of the fact that such molecular composition corresponds to a concentration of SDS in water larger than the critical micelle concentration, it ensures the absence of micelles. Accordingly, the possible formation of ordered clathrate type structures can be attributed only to the capability of SDS to organize water molecules in such a way that, as already mentioned, the methane molecule is driven and then dropped inside the water cage.

3.5.1 Potential Energy Surface

As usual in our studies [209–212], the assemblage of the PES is the first heavy test of GEMS. For large system, the potential is formulated as a sum of intermolecular (V_{inter}) and intramoleclar (V_{intra}) empirical terms calibrated using experimental information and extended *ab initio* calculations.

To this end V_{inter} is further decomposed into non electrostatic (V_{nelec}) and electrostatic (V_{elec}) terms, which are assumed to be independent. V_{nelec} is then expressed in terms of pair interactions between individual atoms or groups of atoms, having a given effective polarizability. Both water and methane have small molecular polarizabilities compared with that of SDS. For this reason, no decomposition was attempted for both of them and single interaction centers, placed on the O and on the C atoms (marked as Ow and Cm) of H_2O and CH_4, respectively, were considered (the successful use of a single interaction center for H_2O [213–218], and CH_4 [202], is already documented in the literature). For the molecular polarizability of the SDS molecule (CH_3-$(CH_2)_{11}$-SO_4Na), instead, a decomposition in effective polarizability values associated with the CH_3-, CH_2-, $(\text{-}SO_4)^-$ and Na^+ groups (whose interaction centers are labelled as C3, C2, SO and Na and assumed to be placed on the C, S and Na atoms, respectively) was made.

ISSN: 2279–8773

This decomposition was applied to the description of both V_{intra} and V_{inter} for SDS-CH_4. On the contrary, the effective polarizabilities associated with the individual atoms of the (-SO_4)$^-$ group were considered to describe the SDS-H_2O interaction to the end of avoiding the collapse caused by the electrostatic attraction when considering two point charges.

Pair interactions acting between centers having an effective polarizability and describing the base components of V_{inter} and V_{intra} are formulated in terms of an Improved Lennard Jones (ILJ) function (see for instance Refs. [216]) as follows:

$$V_{ILJ} = \varepsilon \left[c_1 \left(\frac{r_0}{r} \right)^{\beta + 4.0 \left(\frac{r}{r_0} \right)^2} - c_2 \left(\frac{r_0}{r} \right)^m \right] \tag{3.17}$$

with

$$c_1 = \frac{m}{\beta + 4.0 \left(\frac{r}{r_0} \right)^2 - m}$$

$$c_2 = \frac{\beta + 4.0 \left(\frac{r}{r_0} \right)^2}{\beta + 4.0 \left(\frac{r}{r_0} \right)^2 - m} \tag{3.18}$$

Eq. 3.17 contains four parameters two of which, ε and r_0 (also used in the traditional Lennard Jones (LJ) function), are the well depth and the equilibrium distance (i.e. the point at $V_{ILJ} = -\varepsilon$) of each interaction pair, respectively. They have a specific physical meaning and are transferable among different systems [207,208,216,218] ensuring that all partners are equally well described at both intramolecular and intermolecular level. The third parameter, β, defining the falloff of the repulsion and the relative strength of repulsion and attraction, can be varied depending of the nature of the environment and, therefore, is not completely transferable. The fourth parameter, m, depends on the nature of the attraction and, as usual, m=6 was chosen in the case of a dominant dispersion attraction contribution, while m=4 was chosen when the ion-induced dipole attraction dominates the dispersion one.

The values of the parameters, obtained from the effective polarizability for the considered groups are given in Table 3.4. The Ow-Ow pair parameters and the associated electrostatic contributions used in the present study are those previously tested in the investigation of liquid water (made of non rigid H_2O molecules) [216], while the Ow-Na^+ ones

ISSN: 2279-8773

are the same as those already used in the study of the water solvation of M^+ ions (M=Na,K,Rb) and of M^+-Benzene dimers [214].

In the methane molecule C and H atoms, because of their shared electron pairs, have similar electronegativity. Moreover, the high symmetry of CH_4 and the absence of permanent dipole and quadrupole moments prompts the use of a null effective charge on each atom [219]. Accordingly, the electrostatic energy contribution is calculated only from the charge distribution of SDS and water molecules. The monomer geometry and the charge distribution of water are the ones used before to investigate flexible molecules [216], and correspond to a dipole moment equal to 2.4 D. For SDS the same charge distribution adopted for the MD simulations of micelles formation in water [220], is used.

Then, the CH_4-H_2O interaction ($V_{CH_4-H_2O}$) is formulated using a single pair interaction between the two effective centers Ow and Cm while SDS-CH_4 interaction (V_{SDS-CH_4}) is formulated as,

$$V_{SDS-CH_4} = V_{CH_4-CH_3} + \sum_{i=1}^{11} V_{CH_4-(CH_2)_i} + V_{CH_4-SO_4} + V_{CH_4-Na}. \quad (3.19)$$

On the contrary, the SDS-H_2O interaction is formulated using a different decomposition of the molecular polarizability and including the strong electrostatic effects as follows,

$$V_{SDS-H_2O} = V_{H_2O-CH_3} + \sum_{i=1}^{11} V_{H_2O-(CH_2)_i} + \sum_{i=1}^{4} V_{H_2O-(O)_i} + V_{H_2O-S} + V_{H_2O-Na} + V_{elec} \quad (3.20)$$

where V_{elec} is calculated from the previously mentioned charge distributions of SDS and water.

As to V_{intra}, that of the H_2O molecule is taken from Ref. [216] while that of the SDS molecule, is formulated as a sum of the covalent bond (V_{bnd}), the angle (V_{ang}) and the dihedral (V_{dih}) interaction terms plus the non-covalent contributions. For the V_{bnd} and V_{ang} description the parameters are taken from the AMBER Intramolecular Generalized Force Field [221], while for V_{dih} those of Ref. [220] are adopted. The remaining non-covalent contribution are calculated by adopting the ILJ function and the same decomposition of the SDS polarizability used to describe the SDS-CH_4 intermolecular interaction. The pairs considered are CH_3-CH_2, CH_2-CH_2, CH_3-SO_4, CH_2-SO_4, CH_3-Na, CH_2-Na and SO_4-Na whose related values of the related ILJ function parameters are given in Table 3.4.

Table 3.4: The values of ε (well depth), r_0 (equilibrium distance), β and m parameters used to define the pair interactions in eqs. 1 and 2.

Interaction partners	ε / meV	r_0/ Å	β	m
C2-C2	10.86	3.886	8	6
C2-C3	11.70	3.919	8	6
C2-SO	20.69	4.422	8	6
C3-SO	22.91	4.440	8	6
C3-Ow	10.56	3.850	8	6
C2-Ow	9.85	3.814	8	6
S-Ow	6.96	3.946	8	6
O-Ow	6.96	3.946	8	6
C3-Na^+	2.86	3.595	8	6
C2-Na^+	2.76	3.538	8	6
S-Na^+	1.75	3.742	8	6
O-Na^+	1.75	3.742	8	6
Ow-Na^+	151.89	2.732	6.5	4
Ow-Ow	9.06	3.730	7.5	6
Cm-Na^+	2.91	3.676	8	6
Cm-C3	13.73	3.999	8	6
Cm-C2	12.67	3.968	8	6
Cm-SO	25.75	4.467	8	6
Cm-Ow	13.18	3.868	8	6
Cm-Cm	14.98	4.040	8	6

3.5.2 From dimers to solvation spheres simulations

In order to analyze the validity of the proposed interaction we carried out, first, MD simulations for the separate SDS-CH_4, SDS-H_2O and CH_4-H_2O systems using the DL_POLY program [11]. Following the same procedure of Ref. [207,218] (which involves the evaluation of the interaction energy associated with various geometries of the dimer and the extrapolation of the results at low temperature (0 K)), the equilibrium-like structures for both the SDS-CH_4 and SDS-H_2O systems were determined. Then the obtained equilibrium-like structure of SDS-CH_4 was solvated by merely surrounding them by water molecules. Such solvated SDS-CH_4 configuration was taken as the initial one for MD simulations of the SDS-CH_4-$(H_2O)_n$ system. This ensures that the calculations do not start from initial structures biased to the formation of clathrate cages. In this way, a comparison of the CH_4-$(H_2O)_n$ and the SDS-CH_4-$(H_2O)_n$ MD calculations allows an analysis of the role

ISSN: 2279–8773

played by SDS in promoting the formation of methane hydrates.

The first batches of such calculations were devoted to the SDS-H_2O and to the SDS-CH_4 dimers and were performed by considering the micro-canonical ensemble of particles (NVE), from which the equilibrium-like structures were determined from radial and angular distribution functions calculated at low temperature [202]. A time step of 1 fs and a cut-off radius of 9 $\mathring{A}$, were adopted to keep the variation of total energy (E_{total}) smaller than 10^{-5} meV for all the simulations. After equilibration, during which the system tends to reach the most stable configurations, the MD trajectories for the SDS-H_2O and SDS-CH_4 systems were integrated by conserving E_{total} to allow a statistical analysis. At low temperature the mean configuration energy (E_{cfg}) is close to the equilibrium one.

The MD simulations of the SDS-CH_4 dimer were started from the four limiting initial configurations shown in Figure 3.19. All the simulations carried out at low temperature led, after 0.1 ns of equilibration and an additional 1 ns of simulation, to a configuration having the sulfate head closely located near the methane molecule (see Figure 3.20). Such configuration is easily reached during the equilibration period and becomes, therefore, the geometry of election for starting subsequent simulations. The system, in fact, once equilibrated at the desired temperature, can explore for a sufficient long time those regions of the phase space compatible with the corresponding value of E_{total} (and therefore of T).

As a matter of fact, by carrying out a low temperature analysis of the associated distances and angles distributions, it was possible to locate the geometries at which the potential energy has a minimum (for the sake of clarity we note here that we take as equilibrium-like geometry that of T=5 K, the lowest studied temperature). The results of the simulation coincide with distances and angle distributions associated with the preferred geometry of the dimer (see Figure 3.21 where the upper panel shows the Radial Distribution Histogram (RDH) of methane with respect to the S atom while the lower panel shows the corresponding S-Na-CH_4 Angular Distribution Histogram (ADH)). They indicate that the distribution of the S-CH_4 distance values peak at about 4.63 Å and that of the S-Na-CH_4 angles peaks around of 71.14 °.

The result of the similar investigation performed for the SDS-H_2O dimer (using again NVE statistical ensemble of atoms) are shown in Figure 3.22 (upper panel: the RDH of water with respect to the S atom; lower panel: the corresponding ADH for S-Na-H_2O). The calcu-

ISSN: 2279-8773

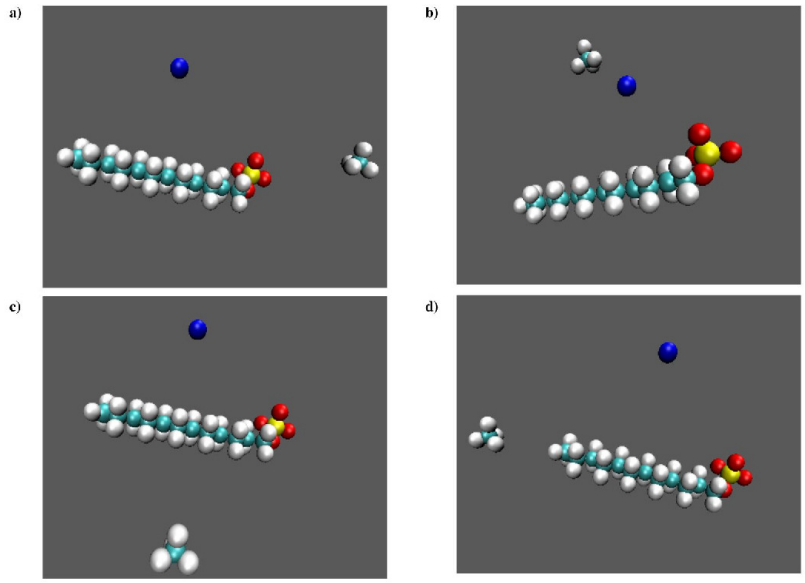

Figure 3.19: The four different initial configurations of the SDS-CH_4 system considered.

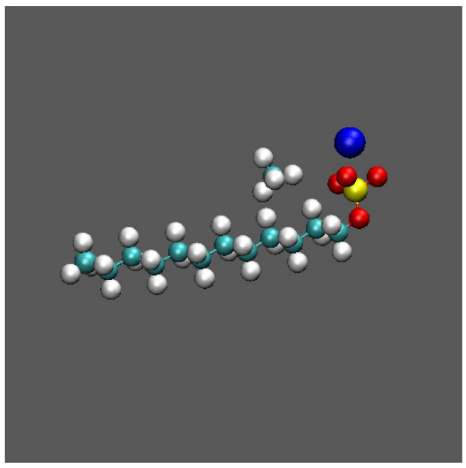

Figure 3.20: SDS-CH_4 equilibrium like structure obtained at T=5 K.

ISSN: 2279-8773

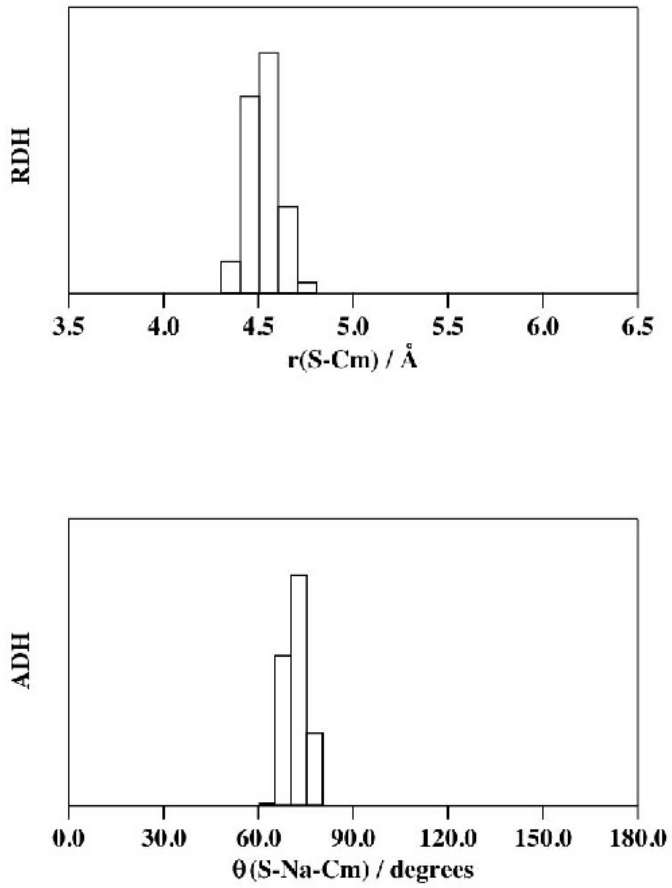

Figure 3.21: S-Cm (top panel) radial distribution histogram and S-Na-Cm (bottom panel) angular distribution histogram obtained at T= 5 K.

lations suggest that the Ow atom of the water molecule is preferentially located about 3.76 Å far from the S atom and forms an angle of 92.70 ° with Na^+.

Additional simulations were performed for both the SDS-$(H_2O)_n$ and the CH_4-$(H_2O)_n$ systems for an increasing number n of non ordered configurations of water molecules using the same conditions adopted for the previous NVE simulations in order to evaluate the amount of energy exchanged associated during the solvation process of SDS and CH_4. In particular, the Radial Distribution Functions (RDF) g(r) [222] ($g(r) = \frac{V}{N^2}\langle\sum_{i\neq j}\delta\left(\mathbf{r}+\mathbf{r}_i-\mathbf{r}_j\right)\rangle$ where r_i, r_j are the positions in the ith and jth molecules, respectively, $\langle\ldots\rangle$ denotes a thermal average, δ is the Dirac delta function, N is the number of atoms considered and V is

ISSN: 2279-8773

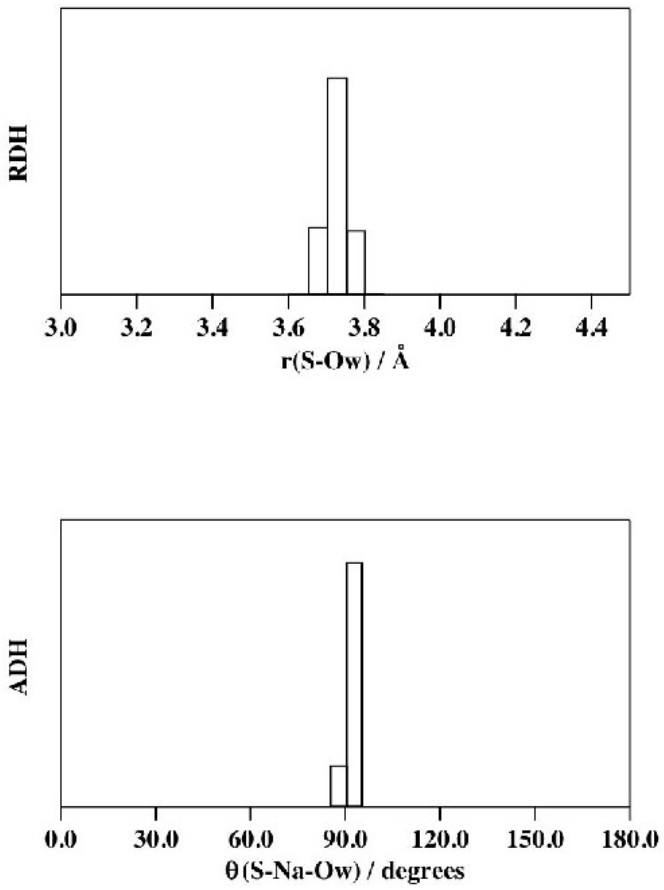

Figure 3.22: S-Ow (top panel) radial distribution histogram and S-Na-Ow (bottom panel) angular distribution histogram obtained at T=5 K.

the volume of the system) of the oxygen atoms of the water molecules with respect both to the sulfur atom and to the C atom for the SDS-$(H_2O)_n$ and the CH_4-$(H_2O)_n$ systems, respectively, were calculated.

The configuration energy per water molecule (E_{cfg}/n(H_2O)) and the preferred radius of the first solvation shell (ISr), derived from the corresponding $g(r)$ functions, are reported in Tables 3.5 and 3.6 for the CH_4-$(H_2O)_n$ and the SDS-$(H_2O)_n$ systems, respectively, for increasing values of n. This provides us with some indications about the solvation energy and the radious (ISr) of the first solvation sphere calculated as the radius (r) associated to the maximum value of $g(r)$.

As the minimum cavity structure able to encapsulate a methane molecule is the 5^{12} one (defined as a pentagonal dodecahedron cage) formed by

Table 3.5: $E_{cfg}/n(H_2O)$ and first solvation shell preferred radius for different numbers of water molecules solvating CH_4.

n	$E_{cfg}/n(H_2O)$ (eV)	ISr (Å)
32	-0.4683	4.05
69	-0.4706	3.88
168	-0.4645	3.78

Table 3.6: $E_{cfg}/n(H_2O)$ and first solvation shell preferred radius for different numbers of water molecules solvating SDS.

n	$E_{cfg}/n(H_2O)$ (eV)	ISr (Å)
30	-0.7531	3.86
63	-0.6156	3.56
170	-0.4749	4.03

20 molecules of water, a minimum of 60 molecules are required to carry out the analysis of the formation of a crystal of methane hydrate (Cubic I). Accordingly, the comparison of the properties of water ensembles containing less than 60 molecules for CH_4-$(H_2O)_n$ with those of the ensembles containing more than 60 molecules gives information about the ability of methane to form structured hydrates as the number of water molecules increases. MD results for the CH_4-$(H_2O)_n$ system do not show any tendencey to form ordered geometries when the number of water molecules increases even if for larger numbers of water molecules the first peak of the radial distribution function moves to lower distances (see Table 3.5). As a matter of fact, it has been observed (see Table 3.5) that the value of $E_{cfg}/$n(H_2O) is almost independent of n. On the contrary, as it can be seen in Table 3.6, an increase of the number of water molecules for the SDS-$(H_2O)_n$ system leads to an increase (less negative value) for $E_{cfg}/$n(H_2O), in agreement with the results of Ref. [223] for SDS-water bulk.

To complete the picture the characteristics of the SDS-CH_4-$(H_2O)_n$ system have been also investigated by considering ensembles of 64 and 168 water molecules. As apparent from the results given in Table 3.7,

ISSN: 2279–8773

Table 3.7: $E_{cfg}/n(H_2O)$ and first solvation shell preferred radius for different numbers of water molecules solvating CH_4.

n	$E_{cfg}/n(H_2O)$ (eV)	ISr (Å)
64	-0.6011	3.56
168	-0.5564	3.78

in which the $E_{cfg}/n(H_2O)$ and ISr (referred to the preferred distances between the O atom of water and the C atom of methane) are given, an increase of the number of water molecules surrounding the SDS-CH_4 dimer leads to an increase of the configuration energy of the system and, at the same time, to an increase of the size of the first solvation shell of CH_4. From a comparison of the results given in Tables 3.5 and 3.7, it can be concluded that the position of the first solvation peak is independent of the SDS molecule when the number of water molecules increases.

3.5.3 The clathrate formation simulation

On the Ground of the pieces of information collected from the simulations discussed above, we carried out realistic MD simulations of single SDS and CH_4 surrounded by an ensemble of non rigid [216] 2000 water molecules under the experimental conditions of temperature and pressure in order to investigate the role played by SDS in the clathrate formation. The simulations were also performed by considering a NpT ensemble of particles (adopting for it cubic boundary conditions) using the same time step and cut-off as in the NVE calculations reported in the previous subsection.

The SDS-CH_4 equilibrium-like geometry (discussed in the previous subsection) was taken as the initial configuration and was solvated using a non ordered ensemble of 2000 water molecules (see Figure 3.23), so as not to favor, as mentioned before, initially solvated CH_4 structures. In order to mimic the experimental conditions of the system near the methane hydrates formation point, a volume of 40 $Å^3$ was chosen and a temperature T= 275 K and a pressure p=8 MPa were adopted. For comparative purposes, similar calculations were performed for a CH_4 molecule immersed in the same pure water bulk at the same tempera-

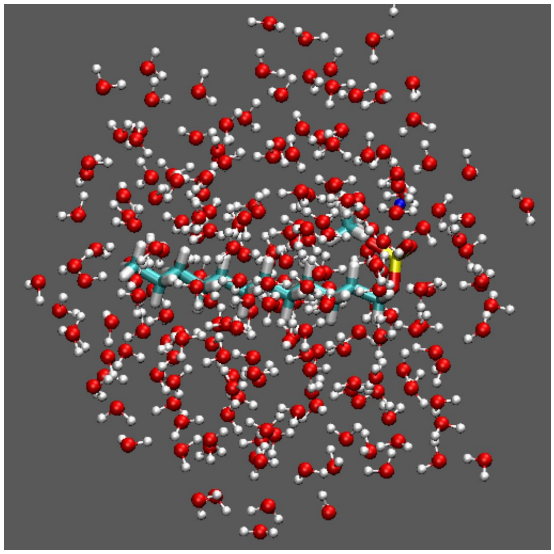

Figure 3.23: Sketch of the SDS-CH_4 equilibrium like structure, obtained at T= 5 K, solvated by an ensemble of 170 water molecules.

ture and pressure conditions (a time step of 1 fs and an equilibration time of 40 ps were used in all the simulations). The MD trajectories were integrated for 2 ns and statistical analyses were performed excluding the equilibration period. Each simulation was performed using a partial integration time of 0.2 ns and a set of continuation runs starting from the final configuration, velocities and forces of the end point of the previous one till reaching the total integration time of 2 ns were launched. Such strategy enabled us to keep under control the dynamic evolution of the studied systems. The calculated densities of the SDS-CH_4 and CH_4 systems in water are 0.9705 g cm^{-3} and 0.9751 g cm^{-3}, respectively.

The analysis of the results obtained from the MD simulation of the SDS-CH_4 system in water pointed out how the folding of a single SDS molecule is able to orient the water molecules to surround CH_4 in a kind of an open cage (see in a sequence the *a*, *b* and *c* screen shots of Figure 3.24 for more details). As apparent from the figure, in fact, the CH_4 slips out of its preferred position near the sulphate (illustrated in Figure 3.20) to end up to be basketed into the water open cage subtended by the SDS (see panel *a* of Figure 3.24). Once the process is completed, the SDS moves away (see *b* and *c* panels of Figure 3.24) and becomes available for replicating the process with other methane molecules. Also in this case, in order to obtain further detailed information on the

ISSN: 2279-8773

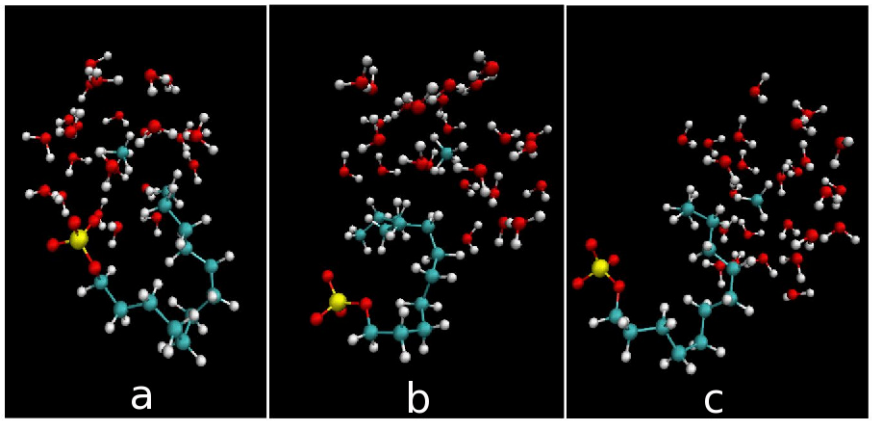

Figure 3.24: Screen shots of folding of the SDS-CH_4 in water taken after a 0.2 *ns* MD simulation using the NpT ensemble.

Table 3.8: Radius and coordination number for methane in the first solvation shell. The MD results, obtained in presence of SDS, are compared with experimental results of Soper et al. [224]

System	MD		Exp. [224]	
	ISr (Å)	N_{CH_4}	ISr (Å)	N_{CH_4}
CH_4-H_2O	3.6	16±1	3.5	16±1
CH_4-H_2O (Hydrate)	3.8	18±1	4.0	21±1

nature of the solvation processes of the methane molecule by a water bulk we calculated for both systems the Cm-OW Radial Distribution Function g(r).

The RDFs (see Figure 3.25) show that the peak indicating the first solvation shell of Methane (ISr) shifts from 3.62 Å to 3.77 Å in going from CH_4-H_2O to SDS-CH_4-H_2O complex system. This shift indicates a modification of the CH_4 solvation shell in presence of SDS. Starting from the ISrs of Methane in the two systems, the respective coordination number (N_{CH_4}) in water has been calculated. A comparison of the values calculated for the two systems is given in Table 3.8 where experimental data are also quoted.

As pointed out by the Table, the calculated values well agree with the experimental ones of Soper et al. [224] obtained using neutron diffrac-

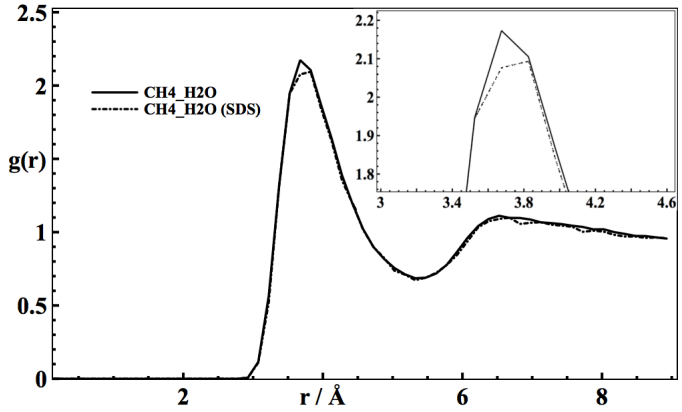

Figure 3.25: A comparison between the Cm-Ow Radial Distribution Functions of the SDS-CH_4-H_2O and the CH_4-H_2O systems. It can be noticed the first peak shifts from 3.62 Å to 3.77 Å.

tion techniques. In such experiments it has been investigated the water ordering around methane during hydrate formation, observing that the hydration sphere around methane in the liquid, changes significantly only once hydrate is formed. In spite of the different conditions used in Ref. [224] with respect to our simulations, the comparison appears to be meaningful, supporting the MD finding of an incompletely formed clathrate structure 5^{12} clearly singled out by Figure 3.26. In that figure half of the 12 pentagons (for a total of 18 molecules forming the structure 5^{12} of the clathrate) are easily detectable with no micelles formation.

3.5.4 HiPEG: the HIgh PErformance Grid

The exploitation of the advantage of using a PES based on an improved formulation of intra- and inter- molecular interaction, utilizing the decomposition of the molecular polarizability, while allowing the building of a road map suited to verify whether or not the formation of micelles hits its limit when tackling realistic treatments of natural systems composed by many particles (atoms and molecules). When considering the present composition of the Grid platform, in fact, one immediately realizes that the size of memory provided and the level of peack velocity offered are insufficient, for example, to extend the use of the quantum techniques to the highest level of theory.

However, the goal of making the packages implementing such tech-

ISSN: 2279-8773

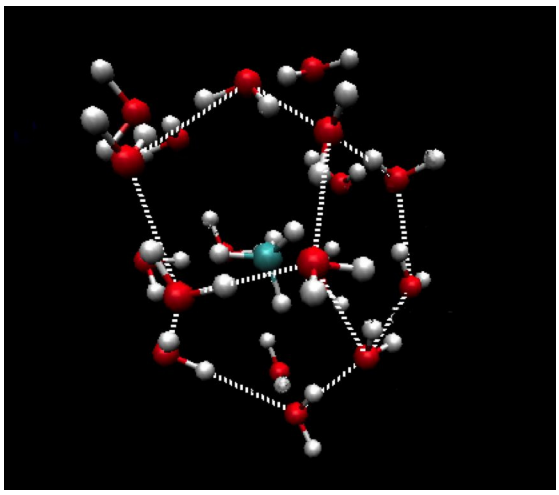

Figure 3.26: Sketch of the solvation sphere of methane at the end of the simulation. From the figure half of the structure 5^{12} of clathrates can be identified.

niques efficient user tools has driven the members of COMPCHEM to foster the evolution of the Grid along the direction of combining a proper usage of High Throughput Computing (HTC) and High Performance Computing (HPC) resources (of both the Super Computer or cluster type based on high speed dedicated networks) for which insufficient effort has been spent up to date to the end of achieving an easy and efficient execution across different computing platforms.

This has motivated, for example, the assemblage of a tool bridging the gap between theHTC and HPC platforms and making the applications interoperate. As a test case CINECA [65] (for HPC) and IGI [84], the Italian Grid Infrastructure (for HTC) have been considered for the interoperable implementation of two applications of the COMPCHEM Virtual Organization (VO) of EGI.

Unfortunately, HPC and HTC infrastructures have developed separately (and sometimes even conflictingly). Related strategies are based in fact on different views of concurrent computing and rely on different types of Middleware. As a matter of fact, they also target two different classes of applications, numerical algorithms and computational approaches. Yet, as previously discussed, on the researchers side a contraposition of HPC and HTC is not what is needed. In many scientific fields researchers need, instead, platforms combining HPC and HTC enabling the accurate modeling of real-like systems as well as virtual

ISSN: 2279-8773

reality simulations based on multi-scale and multi-physics approaches.

To go beyond the present situation we considered as a case study two HTPC skeletons which combine in an iterative fashion an HPC and an HTC computing task. These HTPC s keletons are, indeed, typical of several Computational Chemistry applications of the COMPCHEM VO. However, they are of much more general validity since their ingredients constitute the basic components of many of the presently used computational applications developed not only in Chemistry and Materials Sciences but also in several other disciplines. The two HTPC skeletons are described in some detail below and illustrated by the next two figures.

The first skeleton (HTPC1) is based on a scheme that distributes a large quantity of independent tasks on a HTC platform whose outcomes are passed as input to a strongly coupled treatment implemented on a HPC platform (see Figure 3.27). As shown by the figure, in the first section of the HTPC1 skeleton (that is of the HTC type) an emitter (triangle) generates a (large) number of independent events (circles) each of which provides the input for a HPC highly coupled calculation (square). The outcomes of the distributed HPC tasks are returned (lower layer of arrows) and gathered together by a collector (inverted triangle). In case the information collected is insufficient the sequence is further iterated a certain number of times.

The second skeleton (HTPC2) is based on a scheme that performs a strongly coupled treatment implemented on a HPC platform followed by the distribution of a large quantity of independent tasks to be calculated on a HPC platform (see Figure 3.28). Also in this case the sequence of the two sections is checked against convergence and further iterations are performed (with the associated switch between the two platforms) until either convergence or a maximum number of iterations has been reached.

The systematic construction of Potential Energy Surfaces (and related properties) for complex molecular systems [225] is a typical example of an HTPC1 prototype application. Given its importance in the large majority of chemical problems as an example attaining to this computational scheme, we shall study also the feasibility of constructing a full dimensional PES for a generic molecular system with the purpose of characterizing its main features. In this case, one has to evaluate using a high level ab initio package the potential energy value for a large number of molecular geometries on a HTC platform and then

ISSN: 2279-8773

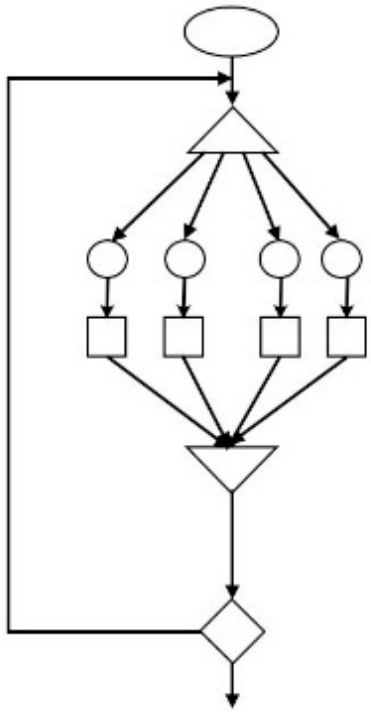

Figure 3.27: Skeleton HTPC1: a HPC computation following a HTC one

from a point representation of the PES the features of the overall interaction (and eventually related molecular properties) are calculated on a HPC platform. Convergence is sought by increasing the number of geometries for which ab initio points are performed.

Suitable examples of the HTPC2 skeleton are the problems amenable to: Calculating kinetic coefficients by quantum mechanical flux correlation functions [226]. As an example attaining to the HTPC2 skeleton we shall explore the case of the calculation of rate coefficients for polyatomic systems by means of flux correlation functions and the MCTDH scheme. In this case, an optimized basis set of multidimensional wavefunctions needs to be generated on a first stage by diagonalizing the full thermal Flux matrix on an HPC platform. Then in the second stage each solution wavefunction is distributed on the Grid for an independent HTC propagation in time. Convergence is checked against the number of multidimensional thermal flux eigenfunctions included in the calculation [225].

ISSN: 2279-8773

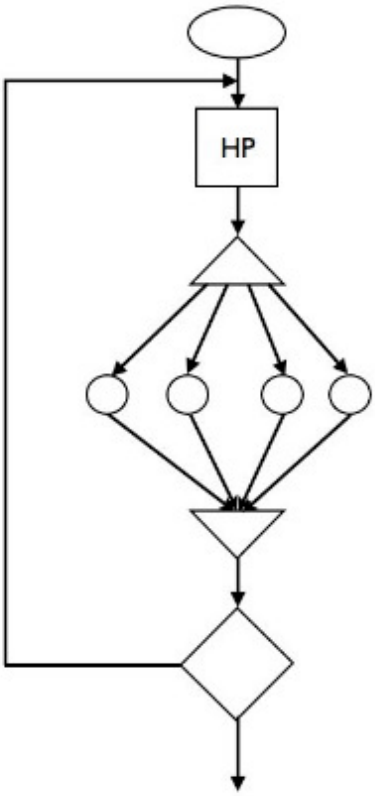

Figure 3.28: Skeleton HTPC2: a HPC computation following a HTC one

3.5.5 Evolution of the work

The main goal of this work is the design and the development of an effective solution of the COMPCHEM problem of combining the use of a Grid infrastructure with a large scale one so as to provide users with both HPC and HTC computing environments. Indeed, the assemblage of the computational skeletons of the two large families of CC applications is a key component of the research activities of the present work and the construction of the associated computational packages will incorporate the fundamental procedures which are building blocks of most of the complex workflows of e-Science applications. At the same time, however, despite the fact that the research activities are driven by the demands of the CC community, their central design principle will consist in keeping the Workflow Engine (WfE) as general purpose as possible, in such a way to accommodate for a wide range

of present and future needs. In doing so, attention will be placed at the definition of a well-suited formalism for process description, which is basically what will be passed as input to the WfE. Another design principle will be the neutrality towards the underlying infrastructure for task execution and data handling, to allow for interoperability with such heterogeneous computing paradigms as the two that this proposal addresses. A layered architecture is foreseen in such a way to abstract from both the Virtual Organization and the infrastructure frameworks, which stand at opposite ends in the proposed scenario, like in the Figure 3.29. Proceeding bottom-up, at the very bottom lies the fabric level, i.e. an unorganized collection of heterogeneous computing and storage resources which are managed by some Local Resource Management Systems (LRMS)s. These resources can be abstracted, within the scope of a single infrastructure by a Grid middleware layer which sits on the LRMS layer providing consistent and homogeneous access to them. Since two different and specific Computing paradigms must be taken into account, an overall Grid Abstraction Layer (GAL) needs to be defined in order to abstract high level functionality such as job submission, data transfer, information retrieval, job tracking & monitoring, etcetera.

ISSN: 2279-8773

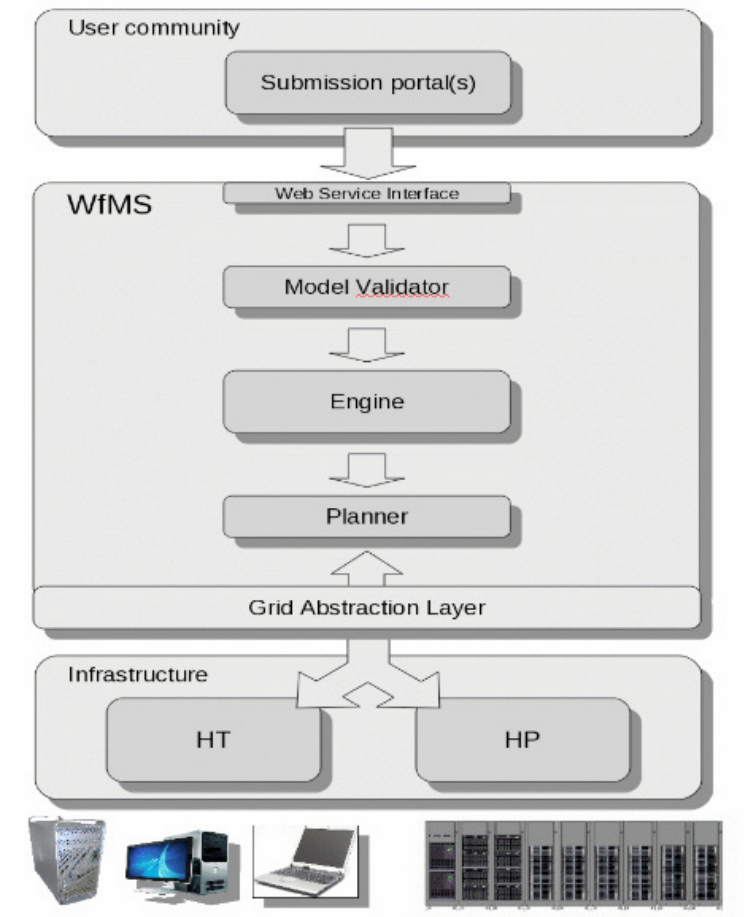

Figure 3.29: The proposed layered architecture

ISSN: 2279-8773

Conclusions

The present thesis is based on the research work developed to investigate the implementation, in a high-throughput computing setting, of a high level of theory simulator dealing not only with Quantum Chemistry approaches but also with quantum mechanics and classical dynamics calculations grafted in a developing for suitable workflows. This was tackled using the computational infrastructure of the EGI Production Grid that support the activities of the Virtual Organization COMPCHEM devoted to deploy and support Computational Chemistry applications.

The research activity has developed by focusing on

- the modeling of complex systems and the design of related simulations,
- the evolution of computer architectures and their impact on elaboration efficiency,
- the organization of computer collaborative environments and tools.

The use cases considered refer to three different class of execution models as follows:

- A Grid execution model for quantum mechanics applications involving three bodies;
- An extension of Grid calculations to Multi-body systems using classical dynamics techniques;
- Implementation of the HIgh Performance Grid extending the realistic treatment of natural systems from few atoms to many body cases enhancing the use of quantum techniques.

In the first execution model the GFIT3C code (a routine performing the global fitting of the potential energy surfaces in triatomic systems)

ISSN: 2279-8773

and ABC application (a quantum mechanical atom-diatom reactive scattering program) have been ported in the EGI Grid Infrastructure and an automatic procedure (workflow) has been developed and tested for them. To achieve this, the use of the CG3Pie high throughput framework allowed us to define event-related dependencies between different applications and execute them simultaneously on a large-scale distributed computing infrastructure.

In the second execution model two codes implementing classical dynamics techniques applied to many body systems have been investigated (GROMACS and DL_POLY) and has been developed for them a set of improved distributed workflows (and related visualization tools) allowing the synergic use of different platforms and the development of better Grid tools.

In the third execution model indications for further development of grid resources are presented and in particular the design and the development of an effective execution model combining an intelligent and interoperable use of the Grid infrastructure with a large scale one so as to provide users with both HPC and HTC computing environments. To do so, a case study of two HTPC skeletons, which combine in an iterative fashion an HPC and an HTC computing task, have been presented.

In support of the validity of the work performed in this thesis, an application of Grid calculations to a real MD study has been presented and the results carried out related to the formation of methane hydrates in sodium dodecyl-sulphate water solutions have been described and analyzed.

ISSN: 2279-8773

Bibliography

[1] J. Yu and R. Buyya, "A taxonomy of scientific workflow systems for grid computing," *SIGMOD Rec.*, vol. 34(3), pp. 44–49, 2005.

[2] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25(5), pp. 528 – 540, 2009.

[3] T. Hey, S. Tansley, and K. Tolle, "The Fourth Paradigm: Data-Intensive Scientific Discovery," *Microsoft Research*, 2009.

[4] "European Grid Infrastructure." http://www.egi.eu.

[5] O. Gervasi, S. Crocchianti, L. Pacifici, D. Skouteris, and A. Laganà, "Towards the Grid design of the Dynamics engine of a molecular simulator," *Lecture Series in Computer and Computational Science*, vol. 7, pp. 1425–1428, 2006.

[6] A. Laganà, A. Riganelli, and O. Gervasi, "On the Structuring of the Computational Chemistry Virtual Organization COMPCHEM," *Lect. Notes Comp. Science*, vol. 3980, pp. 665–674, 2006. http://www.eu-egee.org/compchem.

[7] A. Aguado, C. Tablero, and M. Paniagua, "Global fit of ab initio potential energy surfaces I. Triatomic systems," *Computer Physics Communications*, vol. 108, no. 2-3, pp. 259–266, 1998.

[8] D. Skouteris, J. F. Castillo, and D. E. Manolopulos, "Abc: a quantum reactive scattering program," *Comp. Phys. Comm.*, vol. 133, pp. 128–135, 2000.

[9] "GC3Pie website." http://code.google.com/p/gc3pie/.

[10] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl, "GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation," *J. Chem. Theor. Comp.*, vol. 4, pp. 435–447, 2008.

ISSN: 2279–8773

[11] W. Smith and T. R. Forester, "DL_POLY2: a general purpose parallel molecular dynamics simulation package," *Journal of Molecular Graphics*, vol. 14 (3), pp. 136–141, 1996.

[12] M. Albertí, A. Costantini, A. Laganà, and F. Pirani, "Are Micelles Needed to Form Methane Hydrates in Sodium Dodecyl Sulfate Solutions?," *J. Phys. Chem. B*, vol. 116, pp. 4220–4227, 2012.

[13] D. Gross and H. Carl, *Fundamentals of Queueing Theory*. Wiley, 1998.

[14] M. W. Kirby, *Operational Research in War and Peace: The British Experience from the 1930s to 1970*. Imperial College Press, 2003.

[15] K. A. Atkinson, *An Introduction to Numerical Analysis*. Wiley, 1989.

[16] C. Runge, "Uber die numerische auflösung von differentialgleichungen," *Math. Ann.*, vol. 46, pp. 167–178, 1985.

[17] F. E. Cellier, *Continuous System Modeling*. Springer, 1991.

[18] "JAVA." http://java.sun.com.

[19] E. Y. Chun, H. Chen, and I. Lee, "Web-Based Simulation Experiments," *Proceedings of the 1998 Winter Simulation Conference*, vol. 52, pp. 1649–1654, 1998.

[20] "HTML." www.html.it.

[21] L. Whitman, B. Huff, and S. Palaniswamy, "Commercial Simulation Over the Web," *Proceedings of the 1998 Winter Simulation Conference*, pp. 335–339, 1998.

[22] R. McNab and F. W. Howell, "Using Java for Discrete Event Simulation," *Engineering Workshop, University of Edinburgh, UK*, pp. 219–228, 1996.

[23] R. S. Nair, J. A. Miller, and Z. Zhang, "Java-Based Query Driven Simulation Environment," *Proceedings of the Winter Simulation Conference*, pp. 786–793, 1996.

[24] K. J. Healy and S. R. A. Kilgore, "A java-based process simulation language," *Proceedings of the Winter Simulation Conference*, pp. 475–482, 1997.

[25] D. Knuth, *Electronic Computers Within The Ordnance Corps*. Addison-Wesley, 1997.

[26] J. von Neumann, *Various techniques used in connection with random digits*. U.S. Government Printing Office, 1951.

ISSN: 2279–8773

[27] S. Park and K. Miller, "Random Number Generators: Good Ones Are Hard To Find," *Communications of the ACM*, pp. 1192–1201, 1988.

[28] R. Tausworthe, "Random Numbers Generated by Linear Recurrence Modulo Two," *Mathematics and Computation*, vol. 19, pp. 201–209, 1965.

[29] R. Brent, "Uniform random number generators for supercomputers," *Proc. of Fifth Australian Supercomputer Conference, Melbourne*, vol. 19, pp. 704–706, 1992.

[30] T. Matsumoto, M.; Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8 (1), pp. 3–30, 1998.

[31] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. New York: Dover, 1972.

[32] M. Evans, N. Hastings, and B. Peacock, *Statistical Distributions*. Wiley, 2000.

[33] I. Barany and V. Vu, "Central limit theorems for Gaussian polytopes," *The Annals of Probability (Institute of Mathematical Statistics)*, vol. 35 (4), pp. 1593–1621, 2007.

[34] F. A. Haight, *Handbook of the Poisson Distribution*. Wiley, 1967.

[35] R. G. Sargent, "Verification and validation of simulation models," *Proceedings of the 37th conference on Winter simulation*, pp. 130–143, 2005.

[36] O. Balci, "How to assess the acceptability and credi-bility of simulation results," *Proc. 1989 Winter Simulation Conf.*, pp. 62–71, 1989.

[37] S. I. Gass, "Model accreditation: a rationale and process for determining a numerical rating," *European Journal of Operational Research*, vol. 66 (2), pp. 250–258, 1993.

[38] R. G. Sargent, "Verification and validation of simulation models," *Proceedings of the 2007 Winter Simulation Conference*, pp. 124–137, 2007.

[39] L. Schruben, "Establishing the credibility of simulation models," *Simulation*, vol. 34 (3), pp. 101–105, 1980.

ISSN: 2279–8773

[40] S. Takahashi, "General morphism for modeling relations in multimodeling," *Transactions of the Society for Computer Simulation International*, vol. 13(4), pp. 169–178, 1996.

[41] A. J. Majda, *Atmospheric and ocean science provides a rich source of multiscale problems*. Amer. Math. Soc., 2000.

[42] A. Bensoussan, J. Lions, and G. Papanicolaou, *Asymptotic Analysis of Periodic Structures*. North-Holland, Amsterd-New York, 1978.

[43] J. Kevorkian and J. D. Cole, *Multiple Scale and Singular Perturbation Methods*. Springer, New York, 1996.

[44] V. P. Malsov and M. V. Fedoriuk, *Semiclassical Approximation in Quantum Mechanics*. D. Reidel, Dordrecht-Boston, 1981.

[45] J. B. Keller, "Geometrical theory of diffraction," *J. Opt. Soc. Amer.*, vol. 52, 1962.

[46] A. J. Majda, I. Timofeyev, and E. Vanden-Eijnden, "Systematic strategies for stochastic mode reduction in climate," *J. Atmos. Sci.*, 2003.

[47] A. J. Chorin, "Conditional expectations and renormalization," *Multiscale Model. Simul.*, vol. 1, 2003.

[48] K. Wilson *Phys. Rev.*, vol. B4, 1971.

[49] A. Brandt, "Multigrid methods in lattice field computations," *Nuclear Phys. B*, vol. Proc. Suppl. 26, 1992.

[50] E. B. Tadmor, M. Ortiz, and R. Phillips, "Quasicontinuum analysis of defects in crystals," *Phil. Mag.*, vol. A73, 1996.

[51] R. B. Bird, C. F. Curtiss, R. C. Armstrong, and O. Hassager, *Dynamics of Polymeric Liquids, vol. 2: Kinetic Theory*. Wiley, New York, 1987.

[52] F. F. Abraham, J. Q. Broughton, N. Berdstein, and E. Kaxiras, "Spanning the continuum to quantum length scales in a dynamic simulation of brittle fracture," *Europhys. Lett.*, vol. 44 (6), 1998.

[53] E. Vanden-Eijnden, "Numerical techniques for multiscale dynamical systems with stochastic effects," *Comm. Math. Sci.*, vol. 1, 2003.

[54] C. Theodoropoulos, Y. Qian, and I. G. Kevrekidis *Proc. Nat. Acad. Sci.*, vol. U.S.A. 97 (18), 2000.

[55] J. P. Hansen and I. R. McDonald, *Theory of simple liquids*. 2nd Ed., Academic, 1986.

ISSN: 2279–8773

[56] H. Goldstein, *Classical Mechanics*. Addison-Wesley, Massachusetts, 1950.

[57] L. Landau and E. Lifshitz, *Mechanics*. Pergamon Press, 1961.

[58] K. Kempf, *Electronic Computers Within The Ordnance Corps*. The U.S. Army Research Lab., 1961.

[59] G. Sohi, S. Breach, and T. Vijaykumar, " Multiscalar processors," *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pp. 414–425, 1995.

[60] S. Orlando, *Parallelizing and optimizing compiler*. 1994.

[61] F. Quintana, J. Corbal, R. Espasa, and M. Valero, "Adding a vector unit to a superscalar processor," *International Conference on Supercomputing*, pp. 1–10, 1999.

[62] C. Schauble, *Vector Computing: An Introduction*. High Performance Scientific Computing University of Colorado at Boulder, 1995.

[63] M. Flynn, "Very high-speed computing systems," *Proc. of the IEEE*, vol. 54, p. 1901, 1966.

[64] "Cell Microprocessor Briefing. IBM, Sony Computer Entertainment Inc., Toshiba Corp. 7 February 2005." .

[65] "Centro di supercalcolo, Consorzio di 35 Università italiane." http://www.cineca.it.

[66] I. Foster, *Designing and Building Parallel Programs*. Argonne National Laboratory: Addison-Wesley Inc., 1995.

[67] V. Sunderam, "PVM: A framework for parallel distributed computing," *Concurrency: practice and experience*, vol. 2(4), pp. 315–339, 1990.

[68] G. Geist and V. Sunderam, "Network based concurrent computing on the PVM system," *Concurrency: practice and experience*, vol. 4(4), pp. 293–311, 1992.

[69] A. Beguelin, J. Dongarra, G. Geist, R. Manchek, and V. Sunderam, *A user's guide to PVM Parallel Virtual Machine*. Tennessee: Oak Ridge National Laboratory, 1992.

[70] M. Smir, S. Otto, S. Huss-Ledermam, D. Walker, and J. Dongarra, "MPI: The complete reference," *Int. J. of Supercomputer Applications*, vol. 8(3/4), 1994.

[71] "EGI-Inspire project." http://www.egi.eu/about/egi-inspire.

ISSN: 2279–8773

[72] I. Foster and C. Kesselman, *The Grid: Blueprint for a Future Computing Infrastructure*. USA: Morgan Kaufmann Publishers, 1999.

[73] "ARC Middleware." www.nordugrid.org/arc/ .

[74] "gLite website." http://glite.web.cern.ch/glite.

[75] "UNICORE Mddleware." www.unicore.eu.

[76] "The Globus Project." http://www.globus.org.

[77] "EGEE website." http://public.eu-egee.org.

[78] "European Middleware Initiative." http://www.eu-emi.eu.

[79] "dCache home page." http://www.dcache.org/.

[80] "Unified Middleware Distribution." https://wiki.egi.eu/wiki/Middleware.

[81] "Open Grid Services Architecture." http://en.wikipedia.org/wiki/Open_Grid_Services_Architecture.

[82] "XML." www.html.it/xml/.

[83] "EGI Design Study project." http://web.eu-egi.eu/.

[84] "IGI (Italian Grid Infrastructure)." http://www.italiangrid.it/.

[85] J. J. Andreeva, B. B. Gaidioz, J. J. Herrala, G. G. Maier, R. R. Rocha, and P. P. Saiz, "Experiment dashboard: the monitoring system for the lhc experiments," pp. 45–49, 2007.

[86] "GANGA." http://ganga.web.cern.ch/ganga/.

[87] "DIANE." https://wiki.egi.eu/wiki/VO_Services/Services_and_Tools_Portfolio.

[88] "HYDRA." https://documents.egi.eu/public/ShowDocument?docid=327.

[89] "GrelC." http://grelc.unile.it/events.php.

[90] "Kepler project." https://kepler-project.org/.

[91] "P-GRADE." www.p-grade.hu/.

[92] "SOMA2." www.csc.fi/english/pages/soma.

[93] "Taverna." www.taverna.org.uk/.

[94] L. Storchi, C. Manuali, O. Gervasi, G. Vitillaro, A. Laganà, and F. Tarantelli, "Linear algebra computation benchmarks on a model grid platform," *Lect. Notes Comp. Science*, vol. 2658, pp. 297–306, 2003.

[95] "Cannon algorithm." http://en.wikipedia.org/wiki/Cannon%27s_algorithm.

ISSN: 2279-8773

[96] "Fox Algorithm." .

[97] "Strassen algorithm." http://en.wikipedia.org/wiki/Strassen_algorithm.

[98] V. Piermarini, L. Pacifici, S. Crocchianti, A. Laganà, G. D'Agosto, and S. Tasso, "Parallel methods in time dependent approaches to reactive scattering calculations," *Lect. Notes Comp. Science*, vol. 2073, pp. 567–575, 2001.

[99] D. Bellucci, , S. Tasso, and A. Laganà, "Parallel models for a discrete variable wavepacket propagation," *Lect. Notes Comp. Science*, vol. 2331, pp. 908–917, 2002.

[100] V. Piermarini, L. Pacifici, S. Crocchianti, and A. Laganà, "Parallel approaches to the integration of the differential equations for reactive scattering," *Lect. Notes Comp. Science*, vol. 2658, pp. 341–349, 2003.

[101] A. Saracibar, C. Sánchez, E. Garcia, A. Laganà, and D. Skouteris, "Grid computing in time-dependent quantum reactive dynamics," *Lect. Notes Comp. Science*, vol. 5072, pp. 1065–1080, 2008.

[102] "The Chimere Chemistry-Transport Model. A multi-scale model for air quality forecasting and simulation. Institut Pierre-Simon Laplace, INERIS, LISA, C.N.R.S." http://euler.lmd.polytechnique.fr/chimere.

[103] A. Laganà, S. Crocchianti, G. Tentella, and A. Costantini, "The mpi structure od chimere," *Lect. Notes Comp. Science*, vol. 7333, pp. 417–431, 2012.

[104] "NetCDF homepage." http://www.unidata.ucar.edu/software/netcdf.

[105] L. Pacifici, D. Nalli, D. Skouteris, and A. Laganà, "Time dependent quantum reactive scattering on gpu," *Lect. Notes Comp. Science*, vol. 6784, pp. 428–441, 2011.

[106] L. Pacifici, D. Nalli, and A. Laganà, "Quantum reactive scattering calculations on gpu," *Lect. Notes Comp. Science*, vol. 7333, pp. 292–303, 2012.

[107] R. Baraglia, M. Bravi, G. Capannini, A. Laganà, and E. Zambonini, "A parallel code for time independent quantum reactive scattering on cpu-gpu platforms," *Lect. Notes Comp. Science*, vol. 6784, pp. 412–427, 2011.

[108] F. Vella, R. Cefal, A. Costantini, O. Gervasi, and C. Tanci, "Gpu computing in egi environment using a cloud approach (pdf)," 2011.

ISSN: 2279-8773

[109] "Amazon elastic compute cloud (ec2) web site." http://aws.amazon.com/ec2/.

[110] "Eucalyptus website." http://www.eucalyptus.com.

[111] "Nimbus web site." http://www.nimbusproject.org/.

[112] "Open nebula website." http://opennebula.org.

[113] U. Yildiz, A. Guabtni, and A. Ngu, "Business versus scientific workflow: A comparative study," *Technical Report N2009-3, University of California, Department of Computer Science,*, 2009.

[114] T. Hey and A. Trefethen, "Cyberinfrastructure for e-Science," *Science*, vol. 308(6), pp. 817–821, 2005.

[115] E. Deelman and Y. Gil, "Managing large-scale scientific workflows in distributed environments: Experiences and challenges," *e-Science*, p. 144, 2006.

[116] A. Barker and J. van Hemert, "Scientific workflow: A survey and research directions," *International Conference on Parallel Processing and Applied Mathematics*, pp. 746–753, 2008.

[117] M. P. Thomas, J. Burruss, L. Cinquini, G. Fox, D. Gannon, L. Gilbert, G. von Laszewski, K. Jackson, D. Middleton, R. Moore, M. Pierce, B. Plale, A. Rajasekar, R. Regno, E. Roberts, D. Schissel, A. Seth, and W. Schroeder, "Grid portal architectures for scientific applications," *Journal of Physics: Conference Series*, vol. 16(1), p. 2005, 596.

[118] S. Davidson, S. Boulakia, A. Eyal, B. Ludscher, T. McPhillips, S. Bowers, M. Anand, and J. Freir, "Provenance in Scientific-Workflow Systems," *Proceedings of the international conference on Management of data*, 2008.

[119] S. Miles, J. Papay, W. C., P. Lord, C. Goble, and L. Moreau, "Semantic Description, Publication and Discovery of Workflows in myGrid," *Technical Report, University of Southampton*, 2004.

[120] A. Laganà, E. Garcia, A. Paladini, P. Casavecchia, and N. Balucani, "The last mile of molecular reaction dynamics virtual experiments: the case of the OH(N = 1-10) + CO(j = 0-3) reaction," *Faraday Discuss.*, 2012.

[121] "Detailed list of quantum chemistry and solid state physics software." http://en.wikipedia.org/wiki/List_of_quantum_chemistry_and_solid_state_physics_software.

ISSN: 2279-8773

[122] "MOLPRO." www.molpro.net/.

[123] "Dalton." http://www.nsccs.ac.uk/s_dalton.php.

[124] "GAMESS-US." http://www.msg.ameslab.gov/GAMESS.

[125] M. Schmidt, K. Baldridge, J. Boatz, S. Elbert, M. Gordon, J. Jensen, S. Koseki, N. Matsunaga, K. Nguyen, S.Su, T. Windus, M. Dupuis, and J. Montgomery, "General Atomic and Molecular Electronic Structure System," *J. Comput. Chem.*, vol. 14, pp. 1347–1363, 1993.

[126] C. G. Schatz, "Fitting potential energy surfaces," *Lecture notes in Chemistry*, vol. 75, pp. 15–32, 2000.

[127] N. J. Murrell, S. Carter, S. C. Farantos, P. Huxley, and A. J. C. Varandas, *Molecular Potential Energy Functions*. New York, USA: John Wiley & Sons, 1984.

[128] F. J. Aoiz, V. Sáez-Rábanos, B. Martínez-Haya, and T. González-Lezana, "Quasiclassical determination of reaction probabilities as a function of the total angular momentum," *J. Chem. Phys.*, vol. 123, 2005.

[129] D. Skouteris, A. Laganà, G. Capecchi, and H. Werner *Int. J. Quantum Chem.*, vol. 96, pp. 562–567, 2004.

[130] H. Meyer, U. Manthe, and L. Cederbaum *Chem. Phys. Lett.*

[131] U. Manthe, H. Meyer, and L. Cederbaum *J. Chem. Phys.*, vol. 97, pp. 3199–3213, 1992.

[132] D. Ceperley *Rev. Mod. Phys.*, vol. 67, pp. 279–355, 1995.

[133] W. Hase, R. Duchovic, X. Hu, A. Komornicki, K. Lim, D. Lu, G. Peslherbe, K. Swamy, S. Vande Linde, A. Varandas, H. Wang, and R. Wolf, "VENUS96: A General Chemical Dynamics Computer Program," *Quantum Chemistry Program Exchange*, vol. 16, pp. 562–567, 2004.

[134] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[135] O. Gervasi and A. Laganà, "SIMBEX: a Portal for the a priori simulation of crossed beam experiments," *Future Generation Computer Systems*, vol. 20, pp. 703–715, 2004.

[136] C. Manuali, A. Laganà, and S. Rampino, "GriF: A Grid framework for a Web Service approach to reactive scattering," *Computer Physics Communications*, vol. 181, pp. 1179–1185, 2010.

ISSN: 2279–8773

[137] "CMST COST Action D37." http://www.cost.esf.org/index.php?id = 189&action_number=D37.

[138] "CODECS (COnvergent Distributed Environment for Computational Spectroscopy)." http://www.cost.esf.org/domains_actions/cmst/Actions/CM1002.

[139] "Detailed Chemical Kinetic Models for Cleaner Combustion." http://www.ensic.inpl-nancy.fr/cost/.

[140] "A worldwide e-Infrastructure for NMR and structural biology." http://www.wenmr.eu/.

[141] S. Gogouvitis, K. Konstanteli, G. Waldschmidt, S.and Kousiouris, G. Katsaros, A. Menychtas, D. Kyriazis, and T. Varvarigou, "Workflow management for soft real-time interactive applications in virtualized environments," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 193–209, 2012.

[142] S. Rampino, A. Monari, S. Evangelisti, E. Rossi, K. Ruud, and A. Laganà, "A priori modeling of chemical reactions on a grid based virtual laboratory," *Cracow 09 Grid Workshop*, pp. 164–171, 2010.

[143] J. M. Bowman, "Approximate Time Independent Methods for Polyatomic Reactions," *Lecture Notes in Chemistry*, vol. 75, pp. 101–114, 2000.

[144] "AppPot website." http://code.google.com/p/apppot/.

[145] S. Maffioletti, R. Murri, B. Jonen, and S. Scheuring, "Computational workflows with GC3Pie." http://gc3pie.googlecode.com/svn/wiki/posters/euroscipy2011/gc3pie-euroscipy2011.pdf. Poster presented at the EuroSciPy 2011 conference.

[146] J. Dike, *User Mode Linux*. Prentice Hall, April 2006.

[147] "UML website." http://user-mode-linux.sourceforge.net.

[148] A. Costantini, R. Murri, S. Maffioletti, and A. Laganà, "A Grid Execution Model for Computational Chemistry Applications Using the GC3Pie Framework and the AppPot VM Environment," *Lect. Notes Comp. Science*.

[149] "The G95 Project." http://www.g95.org/.

[150] "SMSCG website." http://www.smscg.ch.

[151] S. Rampino, A. Monari, E. Rossi, S. Evangelisti, and A. Laganà, "A priori modeling of chemical reactions on computational

ISSN: 2279-8773

grid platforms: Workflows and data models," *Chemical Physics*, vol. 398, pp. 192–198, 2011.

[152] G. A. Parker, A. Laganà, S. Crocchianti, and R. T. Pack, "A Detailed three dimensional quantum study of the Li + FH reaction," *J. Chem. Phys.*, vol. 102, pp. 1238–1250, 1995.

[153] A. Laganà, A. Ochoa de Aspuru, A. Aguilar, X. Gimenez, and J. M. Lucas, "Threshold effects and reaction barrier in the Li + FH reaction and its isotopic variants," *J. Chem Phys*, vol. 99, pp. 11696–11700, 1995.

[154] V. Piermarini, S. Crocchianti, and A. Laganà *J. Comp. Method in Sciences and Eng.*, vol. 2, pp. 361–367, 2002.

[155] A. Laganà, S. Crocchianti, and V. Piermarini, "Towards a full dimensional exact quantum calculation of the Li + HF reactive cross section," *Lect. Notes Comp. Science*, vol. 3044, pp. 422–431, 2004.

[156] A. Laganà, E. Garcia, and O. Gervasi, "Improved infinite order sudden cross sections for the Li + HF reaction," *J. Chem. Phys.*, vol. 89, pp. 7238–7241, 1988.

[157] M. Baer, E. Garcia, A. Laganà, and O. Gervasi, "An approximate three dimensional quantum mechanical study of the Li + HF → LiF + H reaction," *Chem. Phys. Lett.*, vol. 158, pp. 362–368, 1989.

[158] A. Laganà, X. Gimenez, E. Garcia, and O. Gervasi, "Parallel calculations of approximate 3D quantum cross sections: the Li + HF reaction," *Chem. Phys. Lett.*, vol. 176, pp. 280–286, 1991.

[159] M. Baer, H. Loesch, H. J. Werner, and I. Last, "Integral and differential cross sections for the Li + HF → LiF + H process. A comparison between j_z-quantum mechanical and experimental results," *Chem. Phys. Lett.*, vol. 219, p. 372, 1994.

[160] M. Baer, I. Last, and H. Loesch, "Three-dimensional quantum mechanical study of the Li + HF → LiF + H process: Calculation of integral and differential cross sections," *J. Chem. Phys.*, vol. 101, p. 9648, 1994.

[161] G. G. Balint-Kurti, F. Gogtas, S. Mort, A. Offer, A. Laganà, and O. Gervasi, "A comparison of time-independent and time-dependent quantum reactive scattering Li + HF → LiF + H model calculations," *J. Chem. Phys.*, vol. 99, pp. 9567–9584, 1993.

ISSN: 2279–8773

[162] F. Gogtas, G. G. Balint-Kurti, and A. Offer, "Quantum Mechanical Three-Dimensional Wavepacket Study of the Li + HF → LiF + H Reaction," *J. Chem. Phys.*, vol. 104, p. 7927, 1996.

[163] G. Parker, R. Pack, and A. Laganà, "Accurate 3D quantum reactive probabilities of Li + FH," *Chem. Phys. Lett.*, vol. 202, pp. 75–81, 1993.

[164] A. Laganà, G. Parker, and R. Pack, "Li + FH Reactive cross sections from J=0 accurate quantum reactivity," *J. Chem. Phys.*, vol. 99, pp. 2269–2270, 1993.

[165] A. Laganà, A. Bolloni, and S. Crocchianti, "Quantum isotopic effects and reaction mechanisms: the Li + HF reaction," *Phys. Chem. Chem. Phys.*, vol. 2, pp. 535–540, 2000.

[166] A. Laganà, A. Bolloni, S. Crocchianti, and G. Parker, "On the effect of increasing the total angular momentum on Li + HF reactivity," *Chem. Phys. Lett.*, vol. 324, pp. 466–474, 2000.

[167] O. Hobel, R. Bobbenkamp, A. Paladini, A. Russo, and H. Loesch *Chem. Phys. Lett.*, vol. 6, p. 2168, 2004.

[168] O. Hobel, M. Menendez, and H. Loesch, "The translational energy dependence of the integral reaction cross section for Li + HF($\nu = 0$) → LiF + H," *Phys. Chem. Chem. Phys.*, vol. 3, p. 3633, 2001.

[169] R. Bobbenkamp, H. Loesch, M. Mudrich, and F. Stienkemeier, "The excitation function for Li + HF → LiF + H at collision energies below 80 meV," *J. Chem. Phys.*, vol. 135, pp. 204306–204313, 2011.

[170] S. Rampino, L. Pacifici, and A. Laganà, "Li + FH → LiF + H: a reaction born to behave as barrierless," *J. Chem. Phys.*, vol. (submitted), 2012.

[171] "C. Panse: Cloud Util Plots." http://cran.r-project.org/web/packages/cloudUtil/index.html.

[172] G. Sipos and P. Kacsuk, "Multi-Grid, Multi-User Workflows in the P-GRADE Portal," *Journal of Grid Computing*, vol. 3, pp. 221–238, 2005.

[173] E. Gutiérrez, A. Costantini, J. López Cacheiro, and A. Rodríguez, "G-FLUXO: A workflow portal specialized in Computational Bio-Chemistry," *1st Workshop IWPLS09*, 2009.

ISSN: 2279–8773

[174] D. Thain, T. Tannenbaum, and M. Livny, *Grid Computing: Making The Global Infrastructure a Reality*. Fran Berman, Anthony J.G. Hey, Geoffrey Fox editors, John Wiley, NY, 2003.

[175] "GROMACS tutorials." http://md.chem.rug.nl/education/mdcourse/index.html.

[176] "G-FLUXO website." http://gfluxo.cesga.es.

[177] "CSD." http://www.ccdc.cam.ac.uk/products/csd .

[178] "CURL." http://curl.haxx.se .

[179] "PRODRG." http://davapc1.bioch.dundee.ac.uk/prodrg .

[180] "ITP." http://www.gromacs.org/Documentation/File_Formats/itp_File.

[181] "Jmol." http://jmol.sourceforge.net.

[182] "PDB." http://www.wwpdb.org.

[183] W. Smith, "Molecular dynamics on hypercube parallel computers," *Comp. Phys. Comm.*, vol. 62, pp. 229–248, 1991.

[184] A. Costantini, A. Laganà, L. Pacifici, and O. Gervasi, "An alternative distribution model for the Molecular Dynamics study of liquid Propane on a grid platform," *Computational Science and Applications*, pp. 433–440, 2007.

[185] K. a. Kvenvolden *Rev. Geophys.*, vol. 31, p. 173, 1993.

[186] E. a. Hammerschmidt *Ind. Eng. Chem.*, vol. 26, p. 851, 1934.

[187] I. Chatti, A. Delahaye, L. Fournaison, and J. a. Petitet *Ener. Conv. Mang.*, vol. 46, p. 1333, 2005.

[188] E. J. a. Sloan *Nature*, vol. 426, p. 353, 2003.

[189] J. Vatamanu and P. a. Kusalik *J. Phys. Chem. B*, vol. 112, p. 2399, 2008.

[190] C. Koh and D. a. Sloan *AIChE*, vol. 53, p. 1636, 2007.

[191] S. Alavi and T. a. Woo *J. Chem. Phys.*, vol. 126, p. 044703, 2007.

[192] J. Zhang, C. Lo, P. Somasundaran, A. Lu, A. Couzis, and J. a. Lee *J. Phys. Chem. C*, vol. 112, p. 12381, 2008.

[193] Y. Okano and K. a. Yasuoka *J. Chem. Phys.*, vol. 124, p. 024510, 2006.

[194] A. Vysniauskas and P. a. Bishnoi *Chem. Eng. Sci.*, vol. 38, p. 1061, 1983.

ISSN: 2279-8773

[195] G. Jiang, T. Yunzhong, F. Ning, L. Zhang, B. Dou, and X. Wu, *Proceedings of the 6th International Conference on Gas Hydrates (ICGH 2008)*. 2008.

[196] J. Zhang, S. Lee, and J. a. Lee *Ind. Eng. Chem. Res.*, vol. 46, p. 6353, 2007.

[197] Y. Zhong and R. a. Rogers *Chem. Eng. Sci.*, vol. 55, p. 4175, 2000.

[198] P. Di Profio, S. Arca, R. Germani, and G. a. Savelli *Chem. Eng. Sci.*, vol. 60, p. 4141, 2005.

[199] A. Costantini, A. Laganà, F. Pirani, A. Maris, and W. a. Caminati *Lect. Notes Comp. Science*, vol. 3480, p. 1046, 2005.

[200] A. Costantini, A. Laganà, and F. a. Pirani *Lect. Notes Comp. Science*, vol. 3980, p. 738, 2006.

[201] A. Costantini and A. a. Laganà *Lect. Notes Comp. Science*, vol. 5072, p. 1052, 2008.

[202] A. Costantini, M. Albertí, F. Pirani, and A. a. Laganà *Int. J. Quant. Chem.*, vol. 112, pp. 1810–1817, 2012.

[203] F. Pirani, M. Albertí, A. Castro, M. Moix, and D. a. Cappelletti *Chem. Phys. Lett.*, vol. 394, p. 37, 2004.

[204] M. Albertí, A. Castro, A. Laganà, F. Pirani, M. Porrini, and D. a. Cappelletti *Chem. Phys. Lett.*, vol. 392, p. 514, 2004.

[205] M. Albertí, A. Castro, A. Laganà, M. Moix, F. Pirani, D. Cappelletti, and G. a. Liuti *J. Phys. Chem. A*, vol. 109, p. 2906, 2005.

[206] M. Albertí, A. Aguilar, J. Lucas, D. Cappelletti, A. Laganà, and F. a. Pirani *Chem. Phys.*, vol. 328, p. 221, 2006.

[207] M. Albertí, L. Pacifici, A. Laganà, and A. a. Aguilar *Chem. Phys.*, vol. 327, p. 105, 2006.

[208] M. Albertí, A. Aguilar, J. Lucas, and F. a. Pirani *J. Phys. Chem. A*, vol. 114, p. 11964, 2010.

[209] F. Huarte-Larrañaga, A. Aguilar, J. Lucas, and M. a. Albertí *J. Phys. Chem. A*, vol. 111, p. 8072, 2007.

[210] M. Albertí, A. Aguilar, J. Lucas, D. Cappelletti, A. Laganà, and F. a. Pirani *Chem. Phys.*, vol. 328, p. 221, 2006.

[211] M. Albertí, A. Aguilar, J. Lucas, A. Laganà, and F. a. Pirani *J. Phys. Chem. A*, vol. 111, p. 1780, 2007.

ISSN: 2279–8773

[212] M. a. Albertí *J. Phys. Chem. A*, vol. 114, p. 2266, 2010.

[213] M. Albertí, A. Aguilar, M. Bartolomei, D. Cappelletti, A. Laganà, J. Lucas, and F. a. Pirani *Lect. Notes Comp. Science*, vol. 5072, p. 1026, 2008.

[214] M. Albertí, A. Aguilar, D. Cappelletti, A. Laganà, and F. a. Pirani *Int. J. Mass Spect.*, vol. 280, p. 50, 2009.

[215] M. Albertí, A. Aguilar, M. Bartolomei, D. Cappelletti, A. Laganà, J. Lucas, and F. a. Pirani *Physica Scripta*, vol. 78, p. 058108, 2008.

[216] N. Faginas Lago, F. Huarte-Larrañaga, and . Albertí, M *Eur. Phys. J. D*, vol. 55, p. 75, 2009.

[217] M. Paolantoni, N. Faginas Lago, M. Albertí, and A. a. Laganà *J. Phys. Chem. A*, vol. 113, p. 15100, 2009.

[218] M. Albertí, N. Faginas Lago, A. Laganà, and F. a. Pirani *Phys. Chem. Chem. Phys.*, vol. 13, p. 8422, 2011.

[219] J. Watson, T. Baker, S. Bell, A. Gann, M. Levine, and R. a. Losick, *Biologia Molecular del gen.* 5th Ed. (Ed Panamericana), 2008.

[220] C. Bruce, M. Berkowitz, L. Perera, and M. a. Forbes *J. Phys. Chem. B*, vol. 106, p. 3788, 2002.

[221] "AMBER." http://ambermd.org/dbase.html.

[222] P. Egelstaff, *An Introduction to the Liquid State*. Oxford University Press, New York, 1992.

[223] K. Schweighofer, U. Essmann, and M. a. Berkowitz *Phys. Chem. B*, vol. 101, p. 3793, 1997.

[224] C. Koh, R. Wisbey, X. Wu, R. Westacott, and A. a. Soper *J. Chem. Phys.*, vol. 113, p. 6390, 2000.

[225] l. Storchi, F. Tarantelli, and A. Laganà, "Computing Molecular energy surfaces on the grid," *Lect. Notes Comp. Science*, vol. 3980, pp. 675–683, 2006.

[226] A. Laganà, N. Faginas Lago, S. Rampino, F. Huarte-Larrañaga, and E. Garcia, "Thermal rate coefficients in collinear versus bent transition state reactions: the N+N2 case study," *Physica Scripta*, vol. 78(5), pp. 58116–58125, 2008.

ISSN: 2279–8773