Virt&l–Comm.3.2012.27

# Report activities of the Research Cluster of the University of Perugia. MPI on the EGI Grid Platform

Alessandro Costantini[1], Osvaldo Gervasi[2], Stefano Crocchianti[3], Leonardo Pacifici[3], and Antonio Laganà[3]

[1] INFN Sez. Perugia - IGI, Italy
*e-mail:* alessandro.costantini@pg.infn.it
[2] Dep. of Math. and Inf., University of Perugia, Italy
*e-mail:* osvaldo@unipg.it
[3] Dep. of Chemistry, University of Perugia, Italy
*e-mail:* {croc,xleo,lag}@dyn.unipg.it

**Abstract.** 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 Perugia. The computational research activity of UNIPG is focused on the codes of the Molecular and Materials Science (MMS) community that in Perugia is made by the members of the Computational Dynamics and Kinetics laboratory, of the Theoretical and Computational Inorganic Chemistry and of the in Silico determination of Pharmacodynamic and Pharmacokinetic Fate of Chemicals of the Department of Chemistry. The following sections are describing the work performed by the research cluster in since the EGI-InSPIRE project started.

## 1 Introduction

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

ISSN: 2279-8773

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.

The University of Perugia (UNIPG) cluster in collaboration with the Virtual Organization (VO) COMPCHEM [1, 2] 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 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 project [3].

## 2 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 [4] The case dealt in some detail, here, is a set of three matrix multiplication algorithms.

### 2.1 Cannon algorithm

Cannon algorithm [5] it was developed in 1969 and still represents a memory efficient version of the following matrix multiplication algorithm:

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

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. 1:

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 $\mathsf{A}$ $i$ positions left and the $j$-th column of $\mathsf{B}$ $j$ positions up.

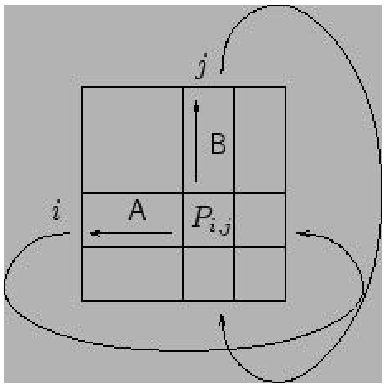

**Fig. 1.** Scheme of the initial step of the Cannon algorithm

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.

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.

## 2.2 Fox algorithm

The Fox algorithm [6] 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.

### 2.3 Strassen algorithm

The Stressen algorithm represent the standard method of matrix multiplication of two $n$ X $n$ matrices takes O(n$^3$) operations. Strassens algorithm [7] 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 communications 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.

## 3 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 inhouse 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 [8] and the time independent [9, 10]) methods with application to the $N+N_2$ system [11]. 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. 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 [12] 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 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. 1 and 2 show in a more quantitative way the percentages plotted in Fig. 2. 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.

**Table 1.** 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.** 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 |

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
- 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.

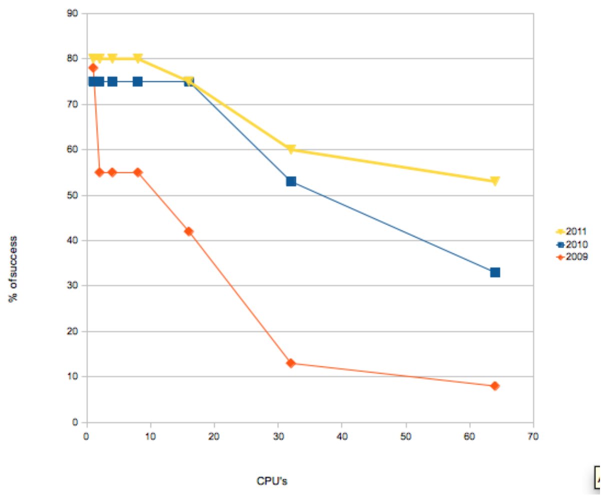

**Fig. 2.** Visual representation of the statistical analysis plotting the percentage of successful jobs against the different years.

## 4 CHIMERE multi scale Eulerian model

The highest level of complexity we tackled was the MPI structuring on the Grid of the package Chimere [13]. 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.

### 4.1 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. 3). The number of needed workers will be, therefore, *nzdoms* x *nmdoms*.

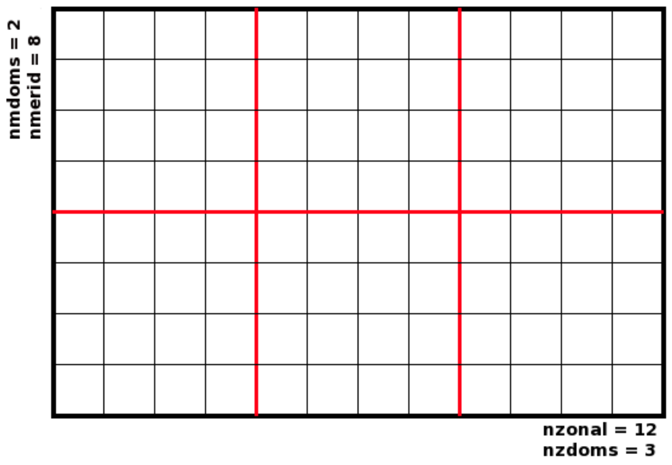

**Fig. 3.** Subdomains division in Chimere

In order to effectively exploit concurrency and achieve significant computing throughput on the Grid a proper distribution model was adopted [14]. 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 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 [15] 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.

### 4.2 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. 4, 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. 5. 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 the cells are evenly divided among the subdomains. This makes the remainder of the division to be unevenly partitioned among the various subdomains.

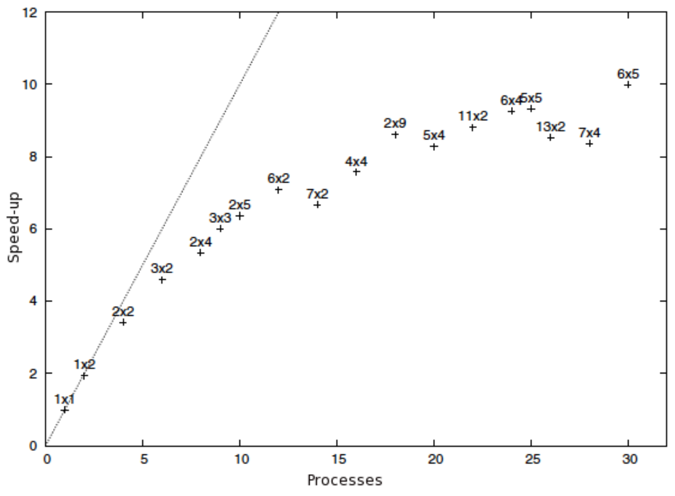

**Fig. 4.** Best speed up value plotted as a function of the number of processes and the corresponding nzdoms x nmdoms combination of values are quoted above the symbol. The dotted line indicates the ideal trend

## 5 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,

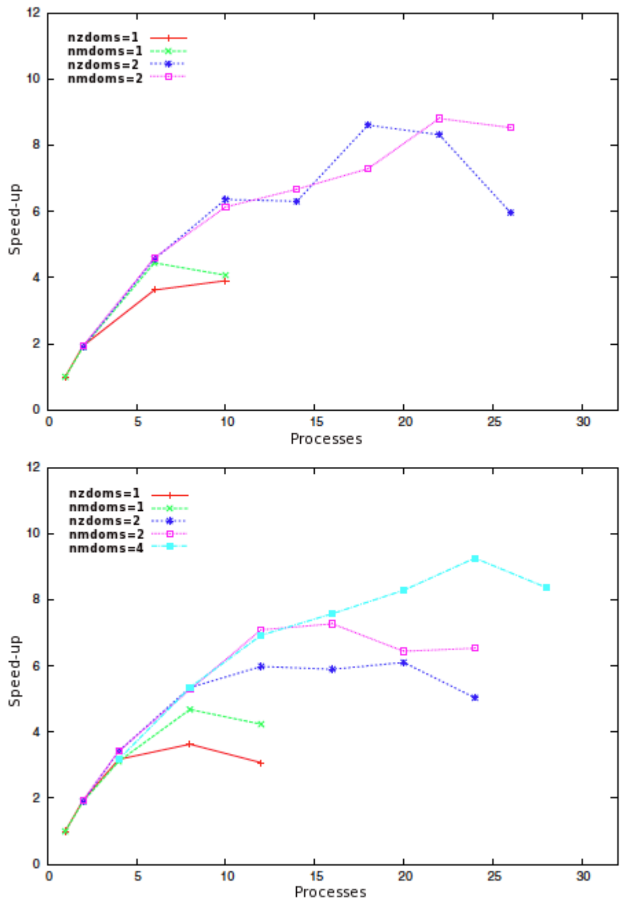

**Fig. 5.** Plot of the speedup as a function of the number of processes when the last node is fully occupied (upper panel) and the last node is partially occupied (lower panel). In the two graphs constant nzdoms or nmdoms series have been evidenced

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 [16–18]. 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 our group [19], 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 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 [20] 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 [21–23]. 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 [21] 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.

## 6 Conclusions

The availability of MPI on the virtually unlimited set of CPUs grafted on Grid platforms has shown to be a strong incentive to implement different scientific applications (from simple basic linear algebra algorithms to complex suites of codes) on distributed systems and to develop appropriate distribution models. On this ground the virtual organization (VO) COMPCHEM assembled out of a group of molecular and material sciences laboratories, committed itself to implement their computer codes on the section of the production European Grid Infrastructure available to the VO using MPI. The presented case studies demonstrates the possibility of using the parallel capabilities of the European Grid and can be used as an example for those Communities which are interested in the porting of their parallel applications into the Grid environment.

## Conferences and Workshops

- MPI support: Users view and prospectives: A. Costantini, L. Pacifici, A. Laganà, S. Crocchianti. EGI Technical Forum, Amsterdam, the Nederlands, 14-17th September 2010.
- GPU computing in EGI environment using a cloud approach: Costantini A., Gervasi O., Vella F., Cefalà R., EGI User Forum 2011, Vilnius, Lithuania, 11-14 April 2011
- Vella, F., Cefalà, R.M., Costantini, A., Gervasi, O., Tanci, C., Proceedings of 11th International Conference on Computational Science and Its Applications, 150-155 (2011)
- MPI and parallel code support: A. Costantini, I. Campos, E. Fernández, A. Laganà, J. Walsh. EGI Technical Forum, Lyon, Paris, 19-23th September 2011.
- MPI in EGI: A Simon, G. Borges, A. Costantini, E. Fernández, EGI Technical Forum, Prague, 17th-21st September 2012.

## Acknowledgments

The authors would like to acknowledge the financial support of the EGI-Inspire project (contract number 261323); MIUR PRIN 2008 (contract number 2008 KJX4SN_003); ESA ESTEC (contract number 21790/08/NL/HE); Phys4entry FP7/2007-2013 (contract number 242311); Fondazione Cassa di Risparmio of Perugia and ARPA Umbria.

The research leading to the results presented in this paper has been possible thanks to the Grid resources and services provided by the European Grid Infrastructure (EGI) and the Italian Grid Infrastructure (IGI). For more information, please reference the EGI-InSPIRE paper (http://go.egi.eu/pdnon) and the IGI web server (http://www.italiangrid.it)

## References

1. 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.
2. A. Costantini, C. Manuali, N. Faginas Lago, S. Rampino, and A. Laganà, "COMPCHEM: progress towards GEMS a Grid Empowered Molecular Simulator and beyond," *Journal of Grid Computing*, vol. 8(4), pp. 571–586, 2010.
3. "EGI-Inspire project." http://www.egi.eu/about/egi-inspire.
4. 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.
5. "Cannon algorithm." http://en.wikipedia.org/wiki/Cannon%27s_algorithm.
6. "Fox Algorithm." .
7. "Strassen algorithm." http://en.wikipedia.org/wiki/Strassen_algorithm.

8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. "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.
14. A. Laganà, S. Crocchianti, G. Tentella, and A. Costantini, "The mpi structure od chimere," *Lect. Notes Comp. Science*, vol. 7333, pp. 417–431, 2012.
15. "NetCDF homepage." http://www.unidata.ucar.edu/software/netcdf.
16. 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.
17. L. Pacifici, D. Nalli, and A. Laganà, "Quantum reactive scattering calculations on gpu," *Lect. Notes Comp. Science*, vol. 7333, pp. 292–303, 2012.
18. 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.
19. F. Vella, R. Cefal, A. Costantini, O. Gervasi, and C. Tanci, "Gpu computing in egi environment using a cloud approach (pdf)," 2011.
20. "Amazon elastic compute cloud (ec2) web site." http://aws.amazon.com/ec2/.
21. "Eucalyptus website." http://www.eucalyptus.com.
22. "Nimbus web site." http://www.nimbusproject.org/.
23. "Open nebula website." http://opennebula.org.