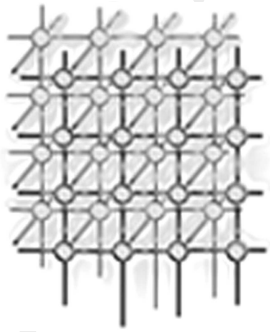
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2006; **18**:1–18
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1114



Performance of computation-intensive, parameter sweep applications on Internet-based Grids of computers: the mapping of molecular potential energy hypersurfaces

S. Reyes[1], C. Muñoz-Caro[1], A. Niño[1,*,†], R. M. Badia[2] and J. M. Cela[2]

[1]*QCyCAR. E. S. Informática. Universidad de Castilla-La Mancha, Paseo de la Universidad, 4. 13071 Ciudad Real, Spain*
[2]*Barcelona Supercomputing Center, Campus Nord UPC, Mòdul D6, Jordi Girona, 1-3. 08034 Barcelona, Spain*

SUMMARY

This work focuses on the use of computational Grids for processing the large set of jobs arising in parameter sweep applications. In particular, we tackle the mapping of molecular potential energy hypersurfaces. For computationally intensive, parameter sweep problems, performance models are developed to compare the parallel computation in a multiprocessor system with the computation on an Internet-based Grid of computers. We find that the relative performance of the Grid approach increases with the number of processors, being independent of the number of jobs. The experimental data, obtained using electronic structure calculations, fits the proposed performance expressions accurately. To automate the mapping of potential energy hypersurfaces, an application based on GRID superscalar is developed. It is tested on the prototypical case of the internal dynamics of acetone. Copyright © 2006 John Wiley & Sons, Ltd.

Received 4 April 2006; Revised 30 June 2006; Accepted 5 July 2006

KEY WORDS: Grid computing; parameter sweep; performance modeling; GRID superscalar; potential energy hypersurfaces; electronic structure calculations

*Correspondence to: A. Niño, QCyCAR. E. S. Informática. Universidad de Castilla-La Mancha, Paseo de la Universidad, 4. 13071 Ciudad Real, Spain.
†E-mail: alfonso.nino@uclm.es

Contract/grant sponsor: Consejería de Educación y Ciencia de la Junta de Comunidades de Castilla-La Mancha; contract/grant number: PBI05-009
Contract/grant sponsor: Ministerio de Educación y Ciencia; contract/grant number: FIS2005-00293
Contract/grant sponsor: Universidad de Castilla-La Mancha
Contract/grant sponsor: Ministry of Science and Technology of Spain; contract/grant number: TIN2004-07739-C02-01

1. INTRODUCTION

Grid computing is a distributed computing model that permits the integration of computational resources from different administrative domains, in order to define a new virtual system. This approach permits to take advantage of individual computational resources, which are usually geographically distributed [1–3]. The Grid is considered the next leap in computer interconnectivity. In the same way as we now share files and facts over the Internet, the Grid lets us share other resources, such as processing power, software or storage space. The connection between the distributed computational resources and the Grid applications is performed by using a layer of middleware software. At present, different middleware tools are available permitting to create and execute software applications on a Grid environment in a transparent way. As examples we have GRID superscalar [4–6], GridWay [7] or P-Grade [8]. In particular, GRID superscalar is a Grid programming framework for Grid unaware applications that allows users to develop applications in a computational Grid. The underlying runtime library is able to automatically parallelize the application. Applications that benefit from GRID superscalar are those composed of tasks, of a certain granularity, which are called several times. GRID superscalar automatically detects the data-dependencies between the tasks of the application, building a task graph on run-time. This allows the user to exploit the inherent parallelism of the application, and concurrent task submission is performed when resources are available. GRID superscalar can use different underlying Grid middlewares as Globus Toolkit 2, Globus Toolkit 4, Ninf-G or ssh/scp.

Many scientific disciplines are considering Grid computing as a new tool. For instance, the astrophysical [9], high-energy physics [10] or computational chemistry [11,12] communities are resorting to Grid computing to perform complex simulations or to process large amounts of data. In particular, Grid computing can be applied to parameter sweep problems, which arise naturally in several scientific and engineering computing fields. Here, we have a system defined by a set of independent parameters. The behavior of the system is analyzed by varying the parameters. This produces a set of jobs that can be computed independently. Depending on the size of the parameter space, the number of jobs can be very large. The problem of the scheduling of these jobs has been considered in heterogeneous environments [13] and, specifically, on computational Grids [14,15]. In this context, a parameter sweep problem of great interest in the physical-molecular field is the evaluation of molecular potential energy hypersurfaces.

Any kind of physical phenomenon related to the internal motion of a molecule depends on the potential energy binding the nuclei of the system. Examples of these phenomena are the conformational distribution of molecules in a vacuum or solution, or the characteristic pattern of energy levels defining the rotovibrational spectrum of a molecule. In particular, the ability to compute an arbitrary potential energy hypersurface will be an invaluable aid in molecular spectroscopic studies. This is so because the potential hypersurface, and its associated molecular structures, permit the construction of rotational and vibrational models. The models serve to interpret and predict the characteristics of rotovibrational molecular spectra. These spectra are chiefly of interest in atmospheric and astrophysical studies to identify molecular species, as well as to determine the conditions under which they are observed [16,17].

For a system of N (nonlinear) nuclei, its relative position is defined by a total of $3N - 6$ parameters. Therefore, the potential energy for the motion of the N nuclei defines a $3N - 6$ dimensional hypersurface. In the framework of the Born–Oppenheimer, adiabatic, approximation, the sum of the quantum mechanical electronic energy plus the electrostatic repulsion energy (for a fixed nuclei)

DOI: 10.1002/cpe

defines the potential for the nuclear motion [18]. Being able to perform quantum-mechanical, electronic structure calculations, the adiabatic approximation provides a parameter sweep approach for mapping a potential energy hypersurface. The technique involves performing a series of electronic structure calculations for different values of the position parameters of interest. From these results, it is possible to write and solve a Hamiltonian for the motion. In this form, the corresponding energy levels (spectroscopically observable) can be obtained, see, for instance, [19]. Formally, it would be possible to deal with the full set of $3N - 6$ parameters, or vibration modes. However, in practice, until now only a few internal motions can be considered simultaneously.

Nowadays, electronic structure methods are available in software packages, and some of them are freely available [20–22]. The quantum-mechanical molecular models implemented in such software tools are very accurate. The predictions they provide for molecular and electronic structure are in very good agreement with the experimental measurements, see, for instance, [23]. However, the computational cost grows with the sophistication of the methodology employed. From such calculations, we can in principle derive a potential energy hypersurface as accurate as needed. The problem is how, and where, to perform the large number of point computations needed for this kind of study. In other words, we are facing a problem halfway between high-performance and high-throughput computing.

Two extreme solutions are possible for a parameter sweep problem such as the mapping of potential energy hypersurfaces. The classical solution is to compute each data point in parallel on a multiprocessor system. Here, the parallelization requires the management of data dependence between the tasks running on independent processors and belonging to the same calculation. We will term this the fine-grained approach. On the other hand, it is possible to compute each point in a single processor, but computing simultaneously as many points as processors. In this case, the calculations are data independent. This last solution will be termed the coarse-grained approach. This approach is well suited for implementation on an Internet-based Grid of computers. For instance, it has been applied to the development of an effective pseudoatom potential for hybrid quantum mechanical–molecular mechanical studies [24]. This potential was derived from a large number of electronic structure calculations obtained by coupling the Gamess electronic structure code [20] to the Nimrod/G Grid distribution tool [25]. However, no performance considerations were taken into account in this work. Due to the difference in communication costs between a dedicated multiprocessor system and an Internet distributed system, the adoption of the fine or coarse-grained approach depends on the relative performance for the problem at hand.

For a given process in a multiprocessor system, the performance depends not only on the execution time, but also on several communication factors. Different performance models have been developed to explicitly account for these communication costs. Thus, the classical LogP approach [26] was based on a Parallel Random Access Machine model. LogP includes the overhead in the transmission and reception of each message, the time gap between consecutive message transmissions or receptions, and the latency in communicating small messages. Variants of LogP have been proposed, reflecting the evolution of the computing architectures. Thus, LogGP accounts for long messages [27]. LoPC and LoGPC are extensions to active messages [28,29], and LoGPS accounts for synchronization costs [30]. Performance models tailored for distributed systems have also been proposed [31–33]. Also, an empirical study has been presented recently on the performance behavior of Grid connected clusters using the NPB set of applications [34].

In this work, we analyze and test the efficiency of the computational Grid paradigm for computationally intensive parameter sweep problems. In particular, we consider the automatic

DOI: 10.1002/cpe

exploration of potential energy hypersurfaces. Performance models are built to compare a (coarse-grained) Internet-based Grid approach with a parallel (fine-grained) approach on a multiprocessor system. Experimental measures are used to test the validity of the proposed mathematical models. In addition, a prototype of an application able to automatically generate a set of molecular structures, run the electronic structure calculations and organize the potential energy results on a computational Grid is presented. The application is tested by considering the torsion of the two methyl groups plus the coupling with the two CCO angles, which defines the molecular frame, in the acetone molecule.

2. PERFORMANCE MODELS

2.1. Fine-grained approach

Let us consider a parallel system (for instance a computer cluster) where one of the processors acts as the master and several processors act as servers. The total time, t_n, used by a parallel program running on n server processors is composed of the time associated to the communication costs plus the time corresponding to the execution of the code. The different factors affecting performance in currently distributed systems have been included by Le and Rejeb [32] in a modified LogGP performance model. Using this model, t_n can be expressed as,

$$t_n = o_{\mathrm{m}} + L + G \cdot S + o_{\mathrm{s}} + h_{\mathrm{m}} + h_{\mathrm{s}} + t_{\mathrm{Seq}} + t_{\mathrm{p}} \tag{1}$$

where o is the master (o_{m}) or server (o_{s}) overhead arising from the communication activities of the processors, L is the transmission latency, G is the message gap per byte, which is the transmission time per unit message, and S is the message length. Both, L and G can be considered independent of the number of processors, n. The term h is the function introduced by Le and Rejeb [32] to describe the overhead cost associated with the use of middleware in the parallelization process in the master (h_{m}) and server (h_{s}). Finally, t_{Seq} and t_{p} represent the time used by the sequential and parallelizable parts of the code, respectively. In Equation (1) we have magnitudes that increase with n, usually those associated to the master. However, other magnitudes decrease with n, such as S, t_{p}, or even those associated to the servers. Therefore, from a general standpoint, Equation (1) can be expressed as

$$t_n = A + B(n) + C(1/n) \tag{2}$$

where A is a constant, and B and C are functions that increase or decrease with n, respectively. When we observe the behavior of actual systems for increasing values of n, we find that the execution time reaches a minimum, and then begins to rise, as a consequence of the increasing role played by communication costs, term $B(n)$ in Equation (2). Therefore, for small or moderate values of n the $C(1/n)$ term dominates. In these conditions we have,

$$(\mathrm{d}B/\mathrm{d}n) \ll (\mathrm{d}C/\mathrm{d}n)) \tag{3}$$

and Equation (2) can be approximated by

$$t_n \cong A' + C(1/n) \tag{4}$$

where A' is a constant. Thus, we have that the total execution time in n processors is composed by an approximately constant term plus a function of $1/n$. In this approach the computation time

DOI: 10.1002/cpe

will be always reduced when increasing n. This represents an overestimation of the performance of parallelization for large n. In other words, we will have an upper (ideal) performance limit for comparison with the Grid approach. This model is simple to use and it will be shown to be useful in real cases, when a limited number of processors are considered.

Since in the conditions defining Equation (4) we have $t_1 > t_n$ for all n, we can establish an analogy with Amdahl's law [35]. Thus, we can assume that t_n can be obtained from t_1 by considering that a proportion, f, of the process is parallelizable. Applying Amdahl's law [35] we obtain

$$t_n = t_1\left(1 + \frac{f \cdot (1-n)}{n}\right) \tag{5}$$

Comparing Equation (5) with Equation (4) we have

$$f = \left(\frac{A' + C[1/n]}{t_1}\right)\frac{n}{(1-n)} \tag{6}$$

Equation (6) shows that f is not a constant, and can be considered as a function of $1/n(f[1/n])$. In other words, $f[1/n]$ does not depend exclusively on the proportion of parallelizable code because it also incorporates communication costs. From Equation (6) we can derive a performance index appropriate for the conditions defining Equation (3), i.e. not very large n values.

The usual performance index is the speedup, defined as $S = t_1/t_n$. Using Equation (6), we can define a computational effort index, E, as the inverse of the speedup:

$$E = S^{-1} = 1 + \frac{f[1/n](1-n)}{n} \tag{7}$$

E, as the speedup, is a non-dimensional magnitude. However, E is normalized in the interval [0,1] for any parallel program. For one server processor the effort is maximum, $E = 1$, decreasing as the number of processors increases. Equation (7) indicates that the highest attainable performance assuming no overheads costs, i.e. the limit for $n \to \infty$, depends on the form of $f[1/n]$.

To give $f[1/n]$ an analytical form, it is useful to expand it in a Taylor series around the $n = 1$ point,

$$f(1/n) = f_1 + \left.\frac{\partial f(1/n)}{\partial(1/n)}\right|_1 (1/n - 1) + \frac{1}{2}\left.\frac{\partial^2 f(1/n)}{\partial(1/n)^2}\right|_1 (1/n-1)^2 + \cdots \tag{8}$$

With Equations (7) and (8), we can give to E the general expression:

$$E = 1 + a\frac{1-n}{n} + b\frac{1-n}{n^2} + c\frac{1-n}{n^3} + \cdots \tag{9}$$

where $a, b, c, \ldots,$ are constants. With respect to Equation (9) two interesting cases can be considered. First, where $f[1/n]$ is a constant (see Equation (8)) we obtain

$$E = 1 + a(1-n)/n \tag{10}$$

Second, considering a linear variation of $f[1/n]$ on $1/n$ (see Equation (8)), we obtain

$$E = 1 + a(n-1)/n + b(n-1)/n^2 \tag{11}$$

Thus, we can derive an analytical expression for E by making some calibration measures followed by a fitting to Equations (10) or (11).

DOI: 10.1002/cpe

In the ideal case where there are no overhead costs, Equation (4) is valid for any n and the minimum possible effort can be obtained as

$$E_\infty = \lim_{n\to\infty} E = 1 - a \tag{12}$$

Equation (12) shows that $1 - a$ is the maximum attainable percentage of time reduction, maxE. The term 'a' can be interpreted as the maximum parallelizability proportion under ideal conditions.

2.2. Coarse-grained approach

Now, a total of m independent tasks will be run in n processors, with $n < m$. In principle, different scheduling schemas for allocating the m process in the n processors are possible [36–40]. We will assume, for the sake of simplicity, that the system is homogeneous and that the computation time of each task is the same, t_1. In these conditions, we can apply a chunk self-scheduling scheme to minimize communication overheads [36]. Thus, we allocate a total of m/n tasks to each of the n processors. Since we are allocating jobs in different processors, the total computing time must include communication costs. Therefore,

$$t_{\text{total}} = \sum_{i}^{m/n} t_i = \frac{m}{n} t_1 + (o_{\text{m}} + L + G \cdot S + o_{\text{s}} + h_{\text{m}} + h_{\text{s}}) \tag{13}$$

where we have used the same nomenclature as in Equation (1). The experience shows that, when considering the same system, communication costs are smaller in Equation (13) than in Equation (1). However, the comparison is not so direct when the fine-grained approach is applied on a dedicated system, and the coarse-grained approach in an Internet-based Grid of computers, due to the differences in latencies, bandwidths and middlewares. In this last case, it is useful to define an F index measuring the ratio of communication to execution costs:

$$F = n(o_{\text{m}} + L + G \cdot S + o_{\text{s}} + h_{\text{m}} + h_{\text{s}})/(mt_1) \tag{14}$$

The F index is formally defined in the interval $(0, \infty)$. Equation (14) shows increasing the number of jobs, m, we decrease the relative weight of the communication costs.

With respect to the electronic structure computations, we must consider that the input data size is small, usually one or two KB in length. On the other hand, and depending on the type of calculation, the results file can have a size of several hundred KB, reaching several hundred MB in the worst case scenario. However, only a limited amount of information is needed when mapping a hypersurface (total energy and vibrational coordinates, at least). Taking into account that a parameter sweep problem involves thousands of points, preselecting locally the return information will reduce communication costs dramatically. In addition, if we consider that typical calculations run at least for a few minutes, and it is not unusual for a calculation to run for a few hours, we can assume that the communication time can be neglected compared to the execution time, i.e. we have a small F factor. Therefore, we will have

$$t_{\text{total}} \cong \frac{m}{n} t_1 \tag{15}$$

Equation (15) shows that the total time depends linearly on the number of calculations and decreases with the inverse of the number of processors. Considering that for one processor $t_{\text{total}} = mt_1$, the

DOI: 10.1002/cpe

computational effort will be

$$E = \frac{1}{n} \tag{16}$$

With the previous assumptions, let us quantify the relative performance of the fine versus the coarse-grained approaches. In the fine-grained approach, for m simultaneous calculations each running on n processors, the total time, t_f, is given by m times t_n. Using Equation (5), we have

$$\lim_{n\to\infty} t_f = \lim_{n\to\infty} mt_1\left(1 + \frac{f[1/n](1-n)}{n}\right) = mt_1(1-C) \tag{17}$$

where C is a constant equal to (see Equation (8))

$$C = f_1 - \left.\frac{\partial f(1/n)}{\partial(1/n)}\right|_1 + \frac{1}{2}\left.\frac{\partial^2 f(1/n)}{\partial(1/n)^2}\right|_1 - \cdots \tag{18}$$

Equation (17) is an underestimation, since the $B(n)$ term of Equation (2) is not taken into account. This term will make the time increase for large n. On the other hand, from Equation (15), and taking into account that the maximum number of processors in the coarse-grained approach equals the number of tasks, we have for the optimal case

$$\lim_{n\to m} t_c = \lim_{n\to m} \frac{m}{n} t_1 = t_1 \tag{19}$$

Comparing equations (17) and (19) we observe that the coarse-grained approach is more efficient as the number of tasks, m, increases. Assuming m to be large (as in the case of the mapping of a potential energy hypersurface, which can involve several thousand points), the coarse-grained approach seems an appropriate solution.

We can determine the relative time gain (RTG) of the coarse versus the fine approach. Thus, dividing t_f–t_c by t_f and converting to percent, we obtain

$$RTG(\%) = 100 \times \left(1 - \frac{1}{n + a(1-n) + b(1-n)/n + \cdots}\right) \tag{20}$$

Equation (20) shows that the RTG is independent of the number of calculations, but depends on the amount of parallelizable code and on the number of processors used. For the simplest case reflected in Equation (11) we obtain

$$RTG(\%) = 100 \times \left(1 - \frac{1}{n + a(1-n)}\right) \tag{21}$$

where 'a' represents the proportion of parallelizability. Equation (21) shows that even with a 90% of parallelizability a great gain is obtained with a small number of processors (as much as a 50% for $n = 10$).

The present results show that a coarse-grained approach to a parameter sweep problem involving several thousand time-consuming independent points, executed on individual processors of an Internet-based Grid of computers, is a highly efficient process. The next step is to analyze experimentally the behavior of this kind of computation on a real Grid system.

DOI: 10.1002/cpe

Marked Proof Ref: cpe1114/33811ae August 11, 2006 Sheet number 7

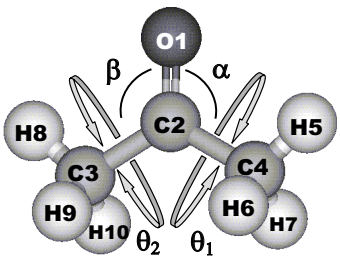


Figure 1. Structure of the acetone molecule. The numbering convention used in this work and the definitions of the θ_1, θ_2, α and β angles are included.

3. METHODOLOGY

As an example of a parameter sweep problem we will use the mapping of molecular potential energy hypersurfaces. For the tests we have selected the acetone molecule. This molecule is considered a prototype for large amplitude vibrations due to the rotation of its two methyl groups. Figure 1 shows the structure and numbering convention of acetone used in this work. We will consider as the main parameters the two methyl torsional angles θ_1, and θ_2, defined by dihedrals O1-C2-C4-H5 and O1-C2-C3-H8, respectively (see Figure 1). As usual, an a-b-c-d dihedral angle corresponds to the angle subtended by planes a-b-c and b-c-d. In addition, we will consider the α and β valence angles (see Figure 1). These two angles define the structure of the CCCO main frame of acetone. Considering the four angles, we are describing the coupling of the two methyl torsions with the acetone molecular frame. Electronic structure calculations on acetone will be carried out at the MP2/6-311G (2d, p) theory level [23]. This level of theory has shown to provide excellent results for the potential energy function for two [41] and three [42] internal degrees of freedom in the molecule of acetaldehyde.

The starting point is a full geometry optimization of acetone at the selected theory level. This defines the zero of relative energy on our hypersurface. Starting from this optimized structure, we perform a scan of the considered angles. In this form, we generate the structures for the mapping of the potential energy hypersurface. For each point on the hypersurface, the values of the angles considered have been kept fixed. All other structural parameters are fully relaxed. All calculations have been performed with the Gamess electronic structure package [20].

The parallel, fine-grained calculations have been carried out on a cluster (named Tales) formed by 12 single-processor Pentium IV, 2.4–3.0 GHz, 1 MB RAM, using a fast Ethernet connection and local storage of intermediate results, as described in reference [43]. The system runs under the cluster configuration and management tool Rocks [44], 3.0.0 version. The details of the Gamess code parallelization can be found in reference [45].

The coarse-grained calculations have been performed both on Tales and on a computational Grid formed by five nodes. The first, Sofocles, is the Grid client. It is a single-processor machine AMD-2.4 GHz from the QCyCAR research group at Ciudad Real, Spain. The next two are two PC clusters of 12 nodes each, also from the QCyCAR research group. The first, Tales, is described above. The second (Hermes) is similar, but consists of Pentium IV, 2.6–2.8 GHz machines. The third node is formed

DOI: 10.1002/cpe

by the Kadesh supercomputing system at the Barcelona Supercomputing Center, Spain. Kadesh is a cluster formed by 128 Power3/375 MHz + 32 Power4/1 GHz nodes with 512 MB of RAM each. The fourth node is a cluster (Popocatepetl) of nine biprocessor 64 bits AMD machines, 1.4 GHz, 1 GB RAM, local storage and ethernet network, placed at the Laboratorio de Química Teórica in the Universidad de Puebla, Mexico. Thus, the Grid implies a transatlantic connection. Hermes and Popocatepetl run under Rocks, version 3.0.0 and 3.2.0, respectively. The Grid runs the Globus toolkit system [46], 2.4.3 version. The heterogeneity of the system is also reflected in the different queuing systems used: Hermes and Tales run OpenPBS [47], Popocatepetl uses Sun Grid Engine [48] and Kadesh LoadLeveler [49]. The independent jobs are sent on the Grid using the 1.6.0 version of GRID superscalar. This version is built on top of Globus Toolkit 2.4.

In order to maintain the consistency of the experimental conditions, all the electronic structure calculations use (independently of the number of processors) the parallel algorithm implemented in Gamess for optimization at the MP2/6-311G (2d, p) level of theory [45]. To just focus on performance issues, the Grid has been used as a dedicated system in the tests.

4. RESULTS AND DISCUSSION

The performance models presented in Equations (10), (11) and (21) can be compared by making some experimental measurements. The tests are carried out considering a potential energy hypersurface for the acetone molecule. Q2

4.1. Determination of Gamess performance at the MP2/6-311G (2d, p) theory level

First, we determine the proportion of parallelizability when working with Gamess at the MP2/6-311G (2d, p) theory level. Thus, we run a full geometry optimization of the acetone molecule varying the number of processors from 1 to 11 in the Tales cluster. Figure 2 collects the results expressed as computational effort, E, relative to a single-processor time of 1448 seconds. The data are fitted to expressions of the type given by Equation (10), for a constant proportion of parallelizability, and Equation (11), for a linear dependence with $(1-n)/n$. The results, including squared correlation coefficient (R^2) and residual sum of squares (RSS), are respectively:

$$E = 1 + 0.7725(n-1)/n \quad \text{with } R^2 = 0.956,\ RSS = 0.023 \tag{22}$$

$$E = 1 + 0.8665(n-1)/n - 0.5558(n-1)/n^2 \quad \text{with } R^2 = 0.990,\ RSS = 0.005 \tag{23}$$

We observe that the fit to the linear case has a higher correlation and a much smaller residual sum of squares. Thus, we will use this case as reference. Therefore, with Equation (23), and from Equation (12), the proportion of effective parallelizability is 86.65%, or approximately 90%.

4.2. Automatic generation of molecular structures

To generate the battery of molecular structures as a function of the $\theta_1, \theta_2, \alpha$, and β angles in our parameter sweep problem, see Figure 1, we need the minimum energy structure of acetone. Previous experimental [50], as well as theoretical [51–53] studies identified the $\theta_1 = \theta_2 = 0^\circ$ as the

DOI: 10.1002/cpe

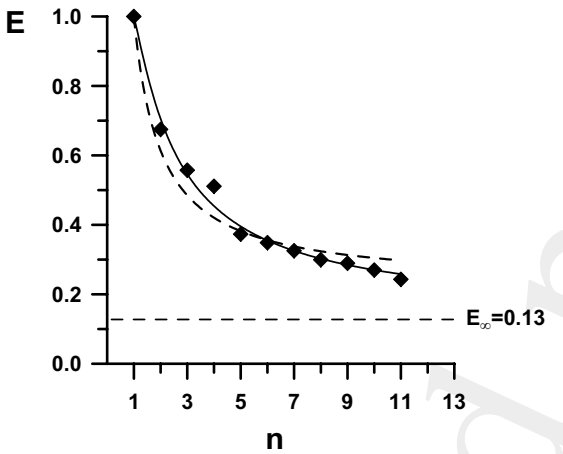


Figure 2. Experimental variation of the E computational effort index with the number of processors, n, in the Tales cluster. The data are obtained for the full geometry optimization of acetone at the MP2/6-311G (2d, p) theory level. The diamonds mark the measured data. The dashed and continuous lines correspond to the fitted curves given by Equations (22) and (23), respectively.

most stable conformer. Thus, we perform a full molecular structure optimization, starting from this configuration, to determine all the structural parameters (distances and angles) in the minimum energy conformation. As expected, our results are $\theta_1 = \theta_2 = 0°$ for this conformer. The other parameters relevant for this study, the α and β angles, are found to be 121.8°. This value is in very good agreement with the 121.9° derived from the experimental data [50]. The battery of points is generated from these starting values.

The results of Niño *et al*. [42], on the coupling of the methyl group and the CCCO molecular frame in acetaldehyde show a maximum variation of 1.5° for the CCO angle. Therefore, we have selected a 2° positive and negative variation around the $\approx$122° α and β equilibrium value. Thus, the α and β values are generated starting from 120° to 124° in increments of 1°. On the other hand, considering the symmetry of acetone [51–53], the torsion of the two methyl groups is described using increments of 20° from 0° to 120° on the θ_1, and θ_2 angles with the restriction $\theta_1 \geq \theta_2$ for $\alpha = \beta$. A total of 1120 different molecular structures are obtained in this way. Generation of the set of points is carried out using nested loops involving the θ_1, θ_2, α and β angles.

4.3. Validation of the relative performance model for the coarse versus the fine-grained parameter sweep approach

Using the eleven processors of the Tales cluster, we can validate Equation (21) considering a variable number of points on the potential energy hypersurface of acetone. These measures are carried out at the cluster level using the OpenPBS queuing system [47]. Thus, sets from 10 to 500 molecular structures, arbitrary selected from the 1120 molecular structures, are considered. The coarse-grained

DOI: 10.1002/cpe

Table I. Experimental results of the fine versus the coarse-grained approaches for different number (m) of molecular structures in the acetone molecule. Each calculation implies a partial molecular structure optimization at the MP2/6-311G (2d, p) theory level. The FG column collects the total time, in seconds, for the calculation of each set of points in the fine-grained approach. The CG column collects equivalent results for the coarse-grained case.

m	FG	CG	RTG (%)
10	2775	1320	52.4
50	17 314	7080	59.1
100	35 772	14 520	59.4
200	75 297	31 080	58.7
300	123 752	49 140	60.3
400	166 350	62 640	62.3
500	209 231	79 380	62.0

approach is applied using the chunk self-scheduling schema [36] with a chunk size of one, i.e. a pure self-scheduling approach. This approach optimizes load balancing. This is specially useful in our case, where the computation time of each job will vary from case to case and where the communication overhead can be neglected compared to the computation time. Thus, we allocate single-processor calculations to free processors until the set of jobs is finished. Table I collects the results. We observe that the RTG value exhibits little variation. The value stabilizes in a 62% for large values of m, the number of data points. This stabilization can be considered as a consequence of the statistical average of the heterogeneity of the system. The observed RTG value of 62% is in good agreement with the 64.8% predicted by Equation (21) using the results of Equation (23).

The previous results validate the performance model proposed in Equation (21). The next step is to apply this approach to the Grid mapping of molecular potential energy hypersurfaces.

4.4. Automating the mapping of potential energy hypersurfaces

The computation of a potential energy hypersurface can be regarded as a single problem, rather than as a sum of individual electronic structure calculations. Thus, we need to implement a unique process able to generate the molecular structures, to perform the individual calculations and to integrate the results. Each calculation needs to run an instance of an electronic structure code with its own input and output file. Our aim is to implement this technique at the Grid level.

From this point of view, a general methodology can be postulated to solve the problem of the generation of molecular potential energy hypersurfaces on a computational Grid. The process is organized in three steps.

DOI: 10.1002/cpe

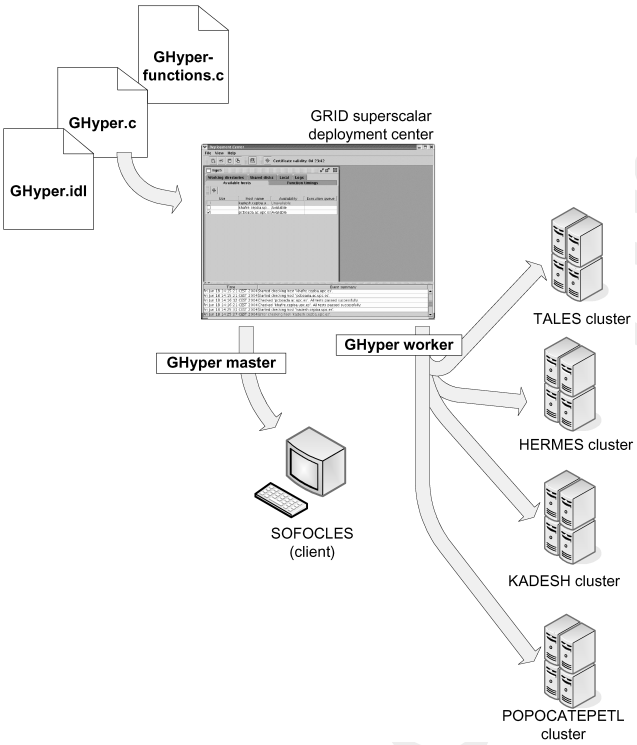


Figure 3. Representation of the files, and the installation process, needed to run the GHyper application with GRID superscalar in a computational Grid.

1. *Molecular structure generation*: this step involves the generation of the set of molecular structures defining the potential energy hypersurface. The result is a large number of input data files.
2. *Electronic structure calculation*: one evaluation with the electronic structure package is executed for each of the data files generated in step 1. Since the output of each of these evaluations is a large output file, a filtering process that obtains the required information (molecular coordinates and total energy) is applied.
3. *Data integration*: the data generated by each of the calculations in step 2 is integrated in a single ASCII file.

While steps 1 and 3 are not computationally intensive, the evaluation of the different data of step 2 would take a long time in a sequential machine. Since the number of points for this kind of job is large (in the current case 1120 electronic structure-independent computations), it is interesting to find a way to parallelize and execute this step in the Grid.

For this *gridification* we use GRID superscalar. The process described by the three previous steps is implemented in a sequential C program, shown in Figure 3 as file GHyper.c. Also, to use GRID

DOI: 10.1002/cpe

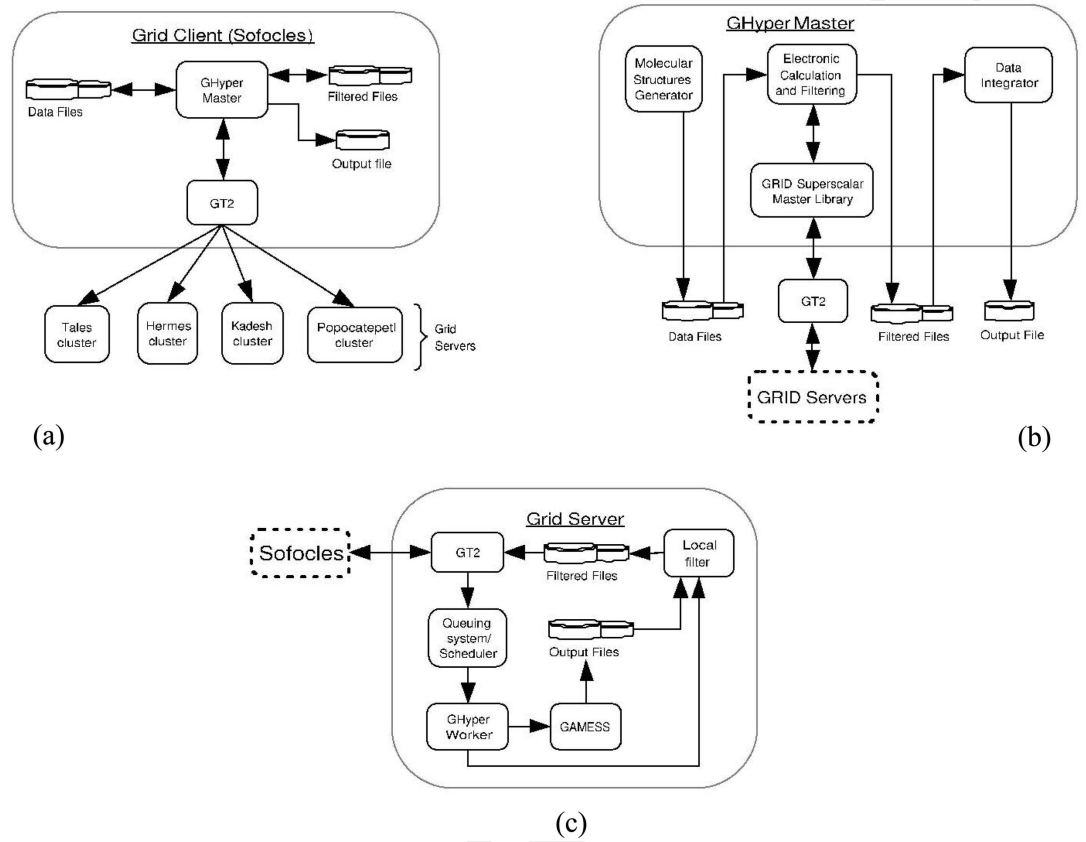


Figure 4. The structure of the application proposed for the automatic computation of molecular potential energy hypersurfaces on a computational Grid. GT2 refers to Globus toolkit version 2.4.3. Case (a) is the high-level description of the whole system. Case (b) is the structure of the GHyper Master process. Case (c) shows the internal working of one of the Grid nodes.

superscalar, we provided the GHyper.idl and GHyper-functions.c files, containing the interfaces and code of the functions to be run in the Grid [4–6]. Two functions are needed here: one for wrapping and calling the electronic structure package, and a second function for the filtering process. The rest of the functions in the application, for example the functions that execute steps 1 and 3, are not listed in the GHyper.idl file, and therefore will be locally executed.

From these three files, and using the GRID superscalar deployment center, two binaries are generated: the GHyper master, which is installed in the client machine; and the GHyper worker, installed on each of the Grid servers machines. The behavior of the whole system is schematically depicted in Figures 4(a)–(c).

DOI: 10.1002/cpe

Table II. Number of jobs (molecular structures) computed on each node of the Grid. The table also collects the total time (in seconds) used to compute the 1120 total jobs. Cases (a) and (b) corresponds to the 32 and 64 nodes distributions, respectively, as described in the text.

	Case (a)	Case (b)
Tales	310	223
Hermes	290	218
Popo	286	262
Kadesh	234	417
Total time	112 500	61 500

The GHyper master starts performing the generation of the input molecular structures files. The program loops over the considered angles and prepares the input files for the Gamess electronic structure package (step 1), see Figure 4(b). Next, the GHyper master continues with step 2. Since the GHyper master is dynamically linked with the GRID superscalar master library, for each electronic structure calculation and filtering process the GRID superscalar broker is queried to decide to which server in the Grid the processes are submitted, see Figure 4(b). Each server in the Grid receives the GRID superscalar requests and the local GHyper worker finally calls the electronic structure calculation (Gamess package) and the filtering function, see Figure 4(c).

All file transfers, scheduling and allocation decisions are performed by the GRID superscalar libraries, in a totally transparent way for the initial GHyper application. Finally, the information (files) generated at the different Grid nodes is sent back by GRID superscalar to the client machine, and integrated in a single output ASCII file (step 3), see Figure 4(a).

The described application is tested considering the four-dimensional problem of acetone. Two different Grid configurations have been used in the tests. The first one uses eight machines (processors) in each cluster of the Grid. Thus, 32 processors are used, distributed as follows: eight on Tales, eight on Hermes, eight on Kadesh and eight on Popocatepetl. The second configuration uses a total of 64 processors: 11 on Tales, 11 on Hermes, 14 on Popocatepetl and 28 processors on Kadesh. The results for both cases are collected in Table II. In absolute terms, we observe that even with 32 processors the time needed to compute the 1120 data points is more than reasonable (31 hours and 15 minutes). When the number of processors doubles, the total time is reduced almost to half (0.55), despite the physical and logical heterogeneity of the system, as a consequence of the linear scaling of the coarse approach, as described by Equations (13) and (15).

These results show that the Grid approach makes possible the computation of the thousands of points needed to map a molecular energy hypersurface in several dimensions. Finally, it is remarkable that the client machine (Sofocles, a 32-bit PC) has been able to handle all the connections on the Grid without trouble. This is a consequence of the asynchrony of the connections needed to send (or receive) data (or results). This asynchrony is favored by the heterogeneity of the system, which makes execution times different from node to node.

DOI: 10.1002/cpe

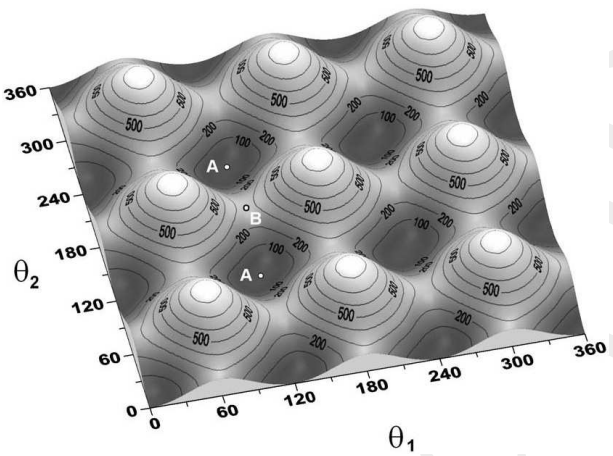


Figure 5. Two-dimensional potential energy hypersurface for acetone as a function of the θ_1, and θ_2 angles. Data obtained in Grid at the MP2/6-311G (2d, p) theory level. Energy in cm^{-1} and angles in degrees. The interval between isocontour lines is 100 cm^{-1}.

4.5. Example of application: Grid evaluation of a potential energy hypersurface for acetone

For the two-dimensional analysis of the acetone hypersurface on the θ_1, and θ_2 angles, our Grid application generates 91 molecular structure files. After submission to the Grid and the collection of the results, the energy data points are used to generate a potential energy hypersurface. This is performed by interpolating with the Kriging method [54] using the Surfer package [55]. Figure 5 shows the resulting hypersurface. The barrier to rotation is obtained as the difference between the minima (for instance, point A in Figure 5) and the first order saddle point connecting two minima (for instance, point B in Figure 5). The value obtained from the interpolated Kriging grid is 271 cm^{-1}, whereas the value obtained directly from the data is 270 cm^{-1}. Table III compares our value to previous theoretical and experimental data. We find that our result, 270–271 cm^{-1}, is within the interval of experimental results, 251–290 cm^{-1}, obtained from fitting experimental rotational and vibrational (torsional) data to different rotovibrational models.

On the other hand, our 1120 four-dimensional results define a database of energy information as a function of the θ_1, θ_2, α, and β angles. Thus, we can gain original information just by selecting the appropriate combination of angles. For instance, this database permits the user to analyze the dependence of the barrier to rotation on the α and β angles. Therefore, we select all the θ_1 and θ_2 data for $\alpha = \beta$ values. For the $\alpha = \beta = 120°$, 121°, 122°, 123° and 124° values we obtain barriers of 231 cm^{-1}, 260 cm^{-1}, 290 cm^{-1}, 322 cm^{-1} and 357 cm^{-1}, respectively. Thus, the barrier, H, increases with the value of the angles. The variation fits extremely well to a quadratic form:

$$H(\mathrm{cm}^{-1}) = 9517.89 - 182.74\alpha - 0.80\alpha^2 \quad \text{with } R^2 = 0.999\,98,\ RSS = 0.019 \tag{24}$$

The increase of the barrier can be explained by an increase of the steric repulsion between the methyl groups, which approach each other when the α and β angles increase.

DOI: 10.1002/cpe

Table III. Barriers to methyl rotation in acetone as determined by different theoretical and experimental methods. All data in cm^{-1}.

Method	Barrier (cm^{-1})
MP2/6-311G (2d, p)[a]	270–271
Rotational data [50]	274
Rotational data [56]	265
Rotational data [57]	272
Torsional data [58]	290
MP2/6-31G (d, p) [53]	267
Rotational + Torsional data [59]	251

[a]This work.

5. CONCLUSIONS

This work considers the suitability of an Internet-based computational Grid for tackling computationally intensive parameter sweep problems. In particular, the mapping of molecular potential energy hypersurfaces is considered. Thus, we compare the computation of large sets of independent jobs in an Internet-based Grid of computers (coarse-grained approach), with the execution on a parallel multiprocessor system (fine-grained approach). The performance models developed show that the performance of the coarse versus the fine-grained approach increases proportionally to the number of jobs. On the other hand, we find that for time-consuming jobs the relative time gain between the coarse and the fine approach is independent of the number of calculations. Even for a 90% of parallelizability, and as few as ten processors, we have a gain of 50%. Thus, the implementation of a parameter sweep problem, such as the mapping of molecular potential energy hypersurface, at the Grid level results a very efficient process. The performance models are contrasted by considering a set of electronic structure calculations on the acetone molecule. The results fit closely the proposed expressions for the parallel performance at the fine-grained level and for the performance of the coarse versus the fine-grained approach.

The practical viability of the mapping of molecular potential energy hypersurfaces on an Internet-based Grid of computers is shown by developing and testing a new Grid application. The application automatically generates molecular structures as a function of the molecular coordinates considered. In addition, the structures are submitted to the Grid for computation in an electronic structure package. The results are filtered in the worker nodes and assembled in the client node. Considering the acetone molecule, a total of 1120 molecular structures are generated. The results show that it is possible to use the Grid to compute thousands of molecular structure data in a more than reasonable time.

As a practical example, the application to acetone permits the mapping map of potential energy hypersurfaces in two and four dimensions. The barrier to methyl rotation obtained from these results is in good agreement with the experimental data. The four-dimensional hypersuface shows that the barrier to methyl rotation increases with the CCO angles. This fact can be attributed to sterical hindrance between the methyl groups. The variation follows a quadratic form extremely well.

DOI: 10.1002/cpe

ACKNOWLEDGEMENTS

The authors wish to thank the Barcelona Supercomputing Center-Centro Nacional de Supercomputación (BSC-CNS) and the Facultad de Ciencias Químicas (Laboratorio de Química Teórica) of the Universidad Autónoma de Puebla (Mexico) for the use of the Kadesh and Popocatepetl clusters, respectively. Q3

REFERENCES

1. Foster I, Kesselman C, Tuecke S. The anatomy of the Grid. Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications* 2001; **15**:200–222.
2. Foster I. THE GRID: Computing without bounds. *Scientific American* 2003; **288**:78–86.
3. Foster I, Kesselmann C (eds.). *The Grid2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann: San Francisco, CA, 2004.
4. Badia RM, Labarta J, Sirvent R, Pérez JM, Cela JM, Grima R. Programming Grid applications with Grid SuperScalar. *Journal of Grid Computing* 2003; **1**:151–170.
5. GRID superscalar Website. http://www.bsc.es/grid [30 July 2006].
6. Sirvent R, Pérez JM, Badia RM, Labarta J. Automatic Grid workflow based on imperative programming languages. *Concurrency and Computation: Practice and Experience* 2006; **18**(10):1169–1186. DOI: 10.1002/cpe.1000.
7. Gridway. http://www.gridway.org/ [30 June 2006].
8. Kacsuk P, Dózsa G, Kovács J, Lovas R, Podhorszki N, Balaton Z. P-GRADE: A Grid programming environment. *Journal of Grid Computing* 2003; **1**:171–197.
9. AstroGrid Project. http://www.astroGrid.org/ [30 June 2006].
10. LHC computing Grid Project. http://lcg.web.cern.ch/LCG/ [30 June 2006].
11. The Extensible Computational Chemistry Environment (ECCE). http://ecce.emsl.pnl.gov/ [30 June 2006].
12. The Computational Chemistry Grid. https://www.gridchem.org/ [30 June 2006].
13. Maheswaran M, Ali S, Siegel HJ, Hensgen D, Freund R. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. *Proceedings of the 8th Heterogeneous Computing Workshop (HCW' 99)*, April 1999.
14. Casanova H, Legrand A, Zagorodnov D, Berman F. Heuristics for scheduling parameter sweep applications in Grid environments. *Proceedings of the 9th Heterogeneous Computing Workshop (HCW00)*, May 2000; 349–363.
15. Huedo E, Montero RS, Llorente IM. Experiences on adaptive Grid scheduling of parameter sweep applications. *Proceedings of the 12th Euromicro Conference on Parallel Distributed and Network-Based Processing (PDP'04)*, 2004; 28.
16. Hollas JM. *Modern Spectroscopy* (3rd edn). Wiley: New York, 1996.
17. *Vibrational-Rotational Spectroscopy and Molecular Structures and Dynamics* (*Advanced Series in Physical Chemistry*, vol. 9), Papoušek D (ed.). World Scientific: Singapore, 1997.
18. Szabo A, Ostlund NS. *Modern Quantum Chemistry*. McGraw-Hill: New York, 1989.
19. Muñoz-Caro C, Niño A, Senent ML. Theoretical study of the effect of torsional anharmonicity on the thermodynamic properties of methanol. *Chemical Physics Letters* 1997; **273**:135–140.
20. Gamess. http://www.msg.ameslab.gov/GAMESS/GAMESS.html [30 June 2006].
21. NWChem. http://www.emsl.pnl.gov/docs/nwchem/nwchem.html [30 June 2006].
22. Dalton. http://www.kjemi.uio.no/software/dalton/dalton.html [30 June 2006].
23. Jensen F. *Introduction to Computational Chemistry*. Wiley: New York, 1998.
24. Sudholt W, Baldridge KK, Abramson D, Enticott C, Garic S. *Applying Grid Computing to the Parameter Sweep of a Group Difference Pseudopotential* (*Lecture Notes in Computer Science*, vol. 3036), 2004; 148–155.
25. Nimrod/G. http://www.csse.monash.edu.au/∼davida/nimrod/index.htm [30 June 2006].
26. Culler DE, Karp RM, Patterson D, Sahay A, Santos EE, Schauser KE, Subramonian R, von Eicken T. LogP: A practical model of parallel computation. *Communications of the ACM* 1996; **39**(11):78–85.
27. Alexandrov A, Ionescu MF, Schauser KE, Scheiman C. LogGP: Incorporating long messages into the LogP model—one step closer towards a realistic model for parallel computation. *Proceedings of the 7th Annual Symposium on Parallel Algorithms and Architecture (SPAA'95)*, Santa Barbara, CA, July 1995; 95–105.
28. Frank MI, Agarwal A, Vernon MK. LoPC: Modeling contention in parallel algorithms. *Proceedings of the 6th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, Las Vegas, NV 18–21 June 1997; 276–287.
29. Moritz CA, Frank MI, LoGPC: Modeling network contention in message-passing programs. *Proceedings of the 1998 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, Madison, WI, 1998; 254–263.

30. Ino F, Fujimoto N, Hagihara K, LogGPS: A parallel computational model for synchronization analysis. *Proceedings of the 8th ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming*, Snowbird, UT, 2001; 133–142.
31. Cameron KW, Ge R. Predicting and evaluating distributed communication performance. *Proceedings of the 2004 ACM/IEEE Conference of Supercomputing (SC2004)*, Pittsburgh, PA, 6–12 November 2004; 43–43.
32. Le TT, Rejeb J. A detailed MPI communication model for distributed systems. *Future Generation Computer Systems* 2006; **22**:269–278.
33. Chronopoulos AT, Penmatsa S, Xu J, Ali S. Distributed loop-scheduling schemes for heterogeneous computer systems. *Concurrency and Computation: Practice and Experience* 2006; **18**(7):771–785.
34. Sokolowski P, Grosu D, Xu C-Z. Analysis of performance behaviors of Grid connected clusters, *Performance Evaluation of Parallel, Distributed and Emergent Systems*, Ould-Khaoua M, Min G (eds.). Nova Science: Hauppauge, NY, 2006.
35. Amdahl GH. Validity of the single processor approach to achieving large scale computing capabilities. *Proceedings of the AFIPS Conference*, Spring Joint Computer Conference, Atlantic City, NJ, 30 April 1967; 483–485.
36. BBN Advanced Computers Inc. *Mach 1000 Fortran Compiler Reference* (revision 1.0 edn), November 1998. BBN Advanced Computer Inc: Cambridge, MA, 1998.
37. Polychronopoulos CD, Kuck DJ, Guided self-scheduling: A practical scheduling scheme for parallel supercomputers. *IEEE Transactions on Computers* 1987; **36**:1425–1439.
38. Hummel SF, Schonberg E, Flynn LE. Factoring: A method for scheduling parallel loops. *Communications of the ACM* 1992; **35**(8):90–101.
39. Tzen TH, Ni LM. Trapezoid self-scheduling: A practical scheduling scheme for parallel compilers. *IEEE Transactions on Parallel and Distributed Systems* 1993; **4**(1):87–98.
40. Chronopoulos AT, Andonie R, Benche M, Grosu D. A class of loop self-scheduling for heterogeneous clusters. *Proceedings of the 3rd IEEE International Conference on Cluster Computing (CLUSTER 2001)*, Newport Beach, CA, 8–11 October 2001; 282–291.
41. Niño A, Muñoz-Caro C, Moule DC. Wagging and torsion vibronic structure in the $T_1 \leftarrow S_0$ electronic spectrum of acetaldehyde. *Journal of Physical Chemistry* 1994; **98**:1519–1524.
42. Niño A, Muñoz-Caro C, Moule DC. Three-dimensional vibrational study of the coupling between the methyl torsion and the molecular frame in the S_0 state of acetaldehyde. *Journal of Physical Chemistry* 1995; **99**:8510–8515.
43. Reyes S, Niño A, Muñoz-Caro C. Customizing clustering computing for a computational chemistry environment. The case of the DBO-83 nicotinic analgesic. *Journal of Molecular Structure: THEOCHEM* 2005; **727**:41–48.
44. http://www.rocksclusters.org/wordpress/ [30 June 2006].
45. Schmidt MW *et al.* General atomic and molecular electronic structure system. *Journal of Computational Chemistry* 1993; **14**:1347–1363.
46. http://www.globus.org/ [30 June 2006].
47. http://www.openpbs.org/ [30 June 2006].
48. http://www.rocksclusters.org/roll-documentation/sge/4.1/ [30 June 2006].
49. Kannan S, Roberts M, Mayes P, Brelsford D, Skovira JF. *Workload Management with LoadLeveler*. IBM RedBooks, IBM, 2001.
50. Swalen JD, Costain CC. Internal rotation in molecules with two internal rotors: Microwave spectrum of acetone. *Journal of Chemical Physics* 1959; **31**:1562–1574.
51. Smeyers YG, Bellido MN. Internal dynamics of nonrigid molecules. I. Application to acetone. *International Journal of Quantum Chemistry* 1981; **19**:553–565.
52. Ozkabak AG, Goodman L. Skeletal flexing during methyl rotation in small dimethyl molecules. *Chemical Physics Letters* 1991; **176**:19–26.
53. Smeyers YG, Senent ML, Botella V, Moule DC. An ab initio structural and spectroscopic study of acetone. An analysis of the far infrared torsional spectra of acetone-h_6 and d_6. *Journal of Chemical Physics* 1993; **98**:2754–2767.
54. Armstrong M. *Basic Linear Geostatistics*. Springer: Berlin, 1998.
55. Golden Software Inc. *Surface Mapping System*, Surfer (V. 8.0), 11 February 2002. Golden Software Inc: Golden, CO, 2002.
56. Peter R, Dreizler H. Das mikrowellenspektrum von aceton im torsionsgrundzustan. *Zeitschrift für Naturforschung A* 1965; **20**:301.
57. Nelson R, Pierce L. Microwave spectrum, structure, and barrier to internal rotation of acetone. *Journal of Molecular Spectroscopy* 1965; **18**:344–352.
58. Groner P, Guirgis GA, Durig JR. Analysis of torsional spectra of molecules with 2 internal C3v rotors. 24. High-resolution far infrared-spectra of acetone-d0, acetone-d3, and acetone-d6. *Journal of Chemical Physics* 1987; **86**:565–568.
59. Groner P. Experimental two-dimensional torsional potential function for the methyl internal rotors in acetone. *Journal of Molecular Structure* 2000; **550–551**:473–479.

DOI: 10.1002/cpe