ELSEVIER

Journal of Molecular Structure: THEOCHEM xx (xxxx) 1–8

THEO CHEM

www.elsevier.com/locate/theochem

Customizing clustering computing for a computational chemistry environment. The case of the DBO-83 nicotinic analgesic

S. Reyes, A. Niño*, C. Muñoz-Caro

Grupo de Química Computacional y Computación de Alto Rendimiento, Escuela Superior de Informática, Universidad de Castilla-La Mancha, 13071 Ciudad Real, Spain

Received 16 February 2005; accepted 18 February 2005

Abstract

Computational chemistry is a quickly evolving field within the computational science discipline. However, the ability to model complex molecular systems and phenomena depends on the availability of computational resources. In this context, clustering computing is becoming a first line tool.

Keywords: Clustering computing; Computational chemistry; DBO-83 nicotinc analgesic

1. Introduction

Clustering computing represents an economic approach to high level, parallel computing. By using off-the-shelf components, a multiprocessor system is built from scratch, with several individual computers integrated in a metasystem. The result is called computer cluster. This kind of metasystem acts like a multiprocessor system, but at a fraction of the cost of a multiprocessor computer. The integration of the different nodes is achieved by using a low cost local area network. Paraphrasing the well known paper by Hargrove et al. [1], clustering computing permits to enter the realms of homemade supercomputing.

This computer approach is made possible by several factors. The first one is the availability of cost-effective computer components in the market, which have now a more than acceptable price/performance ratio. Second, the existence of software publicly available. Thus, we have access to operating systems such as Linux, system and programming tools (including message passing libraries for parallel programming), or user applications (as the electronic structure packages Gamess [2] and NWChem [3], or the Tinker [4] suite of molecular mechanics tools, just in the computational chemistry field). In addition, due to the existence of a growing cluster market, different vendors nowadays sell preassembled cluster systems [5]. However, for the budget of small to medium enterprises and research groups it is more interesting (we mean cheaper) to construct a cluster from standard hardware and software components. In addition, a homemade cluster is easily and cheaply upgraded, when more efficient physical and logical components are available.

The idea of a parallel system build from a set of individual machines is nothing but new. In fact, it is one of the ideas subjacent in the development of Internet. However, the first computer cluster as we understand it today (which was called Beowulf, from the legendary Anglo-Saxon hero that defeats the monster Grendel) was assembled in 1994 by Thomas Sterling and Donald J. Becker at the Center of Excellence in Space Data and Information Sciences (CESDIS) located at the Goddard Space Flight Center. Beowulf was a cluster of 16 DX4 processors running Linux, connected by a 10 Mbps Ethernet network. Since then, Beawulf has become a synonym for any computer cluster build from PCs.

Any computer intensive discipline can benefit from the implicit parallelism of computers clusters. A clear example is computational chemistry. Computational chemistry can be defined as the quantitative computer modelling of chemical phenomena. The basic physical theory used here

* Corresponding author. Tel.: +34 9262 95300; fax: +34 9262 95354.
E-mail addresses: alfonso.nino@uclm.es (A. Niño), quimcom@uclm.es (A. Niño).

0166-1280/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.theochem.2005.02.047

is quantum mechanics, since molecules are nothing but a set of nuclei and electrons, i.e. quantum entities. Therefore, the models and tools based on quantum mechanics are of paramount importance in the field. In this context, electronic structure methods are mandatory tools for the description of molecular phenomena. These models are invaluable for analyzing the behaviour of molecular systems, for instance when modelling the activity of bioactive compounds. The problem is that, to reach quantitative precision, we need extremely large amounts of computational resources. In particular, these electronic structure problems are input–output (I/O) bounded. Then, I/O access represents a limiting factor in the computational process. For these kinds of calculations the use of computer clusters represents an obvious alternative. Therefore, many computational chemistry research groups have turned toward clustering computing as an interesting alternative to traditional high-level workstations.

When first facing the task to build a computer cluster, several questions arise: What physical components must be selected? How the different computers are tied together? How the operating system and the network must be set up? How the user applications must be configured to take advantage of the parallel nature of the cluster? How the different jobs can be managed in a centralized form? In this context, highly automatized Open Source software systems are available for the installation and management of computer clusters [6]. Some of these tools are installed over a Linux distribution or are based in a Linux kernel. Among these, we can highlight Rocks [7]. This tool acts as a true operating system at the cluster level, optimizes the system configuration of the master and computing nodes, and is available for a large number of 32 and 64 bits platforms.

In this paper, we present an organization model for the different steps needed to build and optimize a computer cluster. In the following paragraphs we describe how the different hardware and software components of the system are integrated and configured. In addition, we show how to characterize the behavior of a computer cluster by monitoring the use of computational resources. Also, we present some techniques to build quantitative performance models, in order to optimize the functioning of the system. Finally, we introduce some theoretical models intended to maximize the computer performance for several parallel calculations. The customization of the cluster is exemplified by optimizing the system for electronic structure calculations in the computational chemistry field. The functioning of the system is shown by performing several calculations using the Gamess electronic structure code [2]. In particular, we determine the full conformational space in vacuum, and the most stable structures, of the new nicotinic analgesic DBO-83 in its bioactive, protonated form. Also, we analyze the variation of the internitrogen distance with the conformation, which is one of the index defining the nicotinic pharmacophore.

2. Planning the system

When building a computer cluster, we must consider that one of the machines acts as a master node, whereas the other machines act as slave or client nodes. At this point, it is important to highlight that the whole system follows the client–server model. We have found that most of the problems faced in the configuration and use of a given service or tool, are solved after clarifying which nodes (or node) must act as servers and which nodes (or node) must act as clients. The master node plays essentially a management role. In contrast, the client or computing nodes are devoted to processing tasks.

Apart form this basic structure, there is an efficient model for the kind of applications we are interested in, i.e. ab initio electronic structure calculations of molecular systems. These calculations are input–output bounded, essentially due to the generation of large (several GB for medium size molecules) two-electron integral files. On the other hand, communication among the nodes can be achieved using a network file system (NFS), physically placed on the master node, where the user programs and scratch space can reside. To optimize the performance of the system for our I/O bounded problems, we must reach two goals. First, we must minimize the transfer of large files on the Ethernet network. In second place, it is necessary to overcame the bottleneck associated to the syncronous access to such large files. Both goals can be reached by configuring a scratch space on each computer node, in order each node to perform all the intermediate input–output on the local disk. Thus, only the initial reading of the executable program, the lecture of the data and the writing of the final results involves the NFS. This organization have an immediate advantage. The initial synchronous disk access to a large file is divided in several asynchronous (simultaneous) accesses to smaller files. Therefore, one of the most limiting factors in an electronic structure calculation is cutting down.

3. Configuring a cluster

When organizing a computer cluster, the system can be thought of as a set of increasingly higher abstraction levels, see Fig. 1. The basic level represents the set of physical components and interconnections. From here, using software components, we progress until the point where the user can apply, in a transparent way, parallel software tools. Let us consider briefly each abstraction layer one by one.

3.1. Physical level

At this level of abstraction, we are concerned with the hardware components, and the physical links, needed to build the cluster. Fig. 2 shows the typical cluster architecture, as the one presented here. As indicated above, one of the nodes acts as master. The role of this

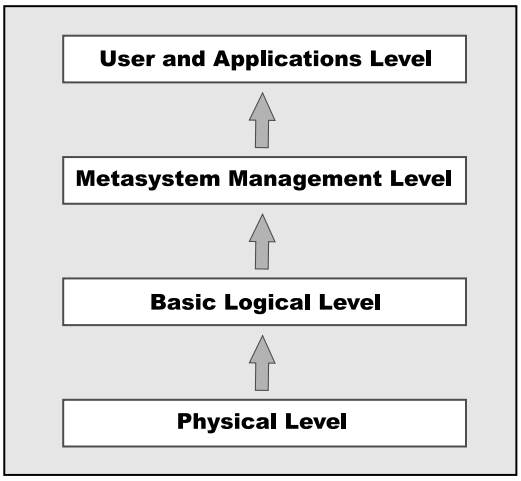

Fig. 1. The layered model of a computer cluster.

computer is to manage the full system, and provide connection with the outside world via internet. The rest of the nodes are used only for performing the cluster tasks, i.e. they play the role of individual processors in a multiprocessor system. This configuration usually leaves the master free from any calculation intensive task. Interconnection between the master and computing nodes is achieved by building a private LAN, for instance, with standard 100 Mbps Ethernet cards and a 100 Mbps switch. A switch is preferred to a hub, because the switch recognizes the MAC, physical, address of the network cards. In this form, information packages are sent only to its target card, optimizing the use of the switch bandwidth.

The processing, or computing, nodes are equipped with just one Ethernet card, whereas the front-end is equipped with two. Each node is connected from the Ethernet card to the switch. The second Ethernet card on the front-end is used to connect this machine to the outworld. Also, in order to install the operating system and any additional tool, a CDROM unit is placed on the master node. The processing nodes need, at most, a floppy disk unit, since the operating system can be installed using a network file system (NFS), exported from the master, or from the LAN using network cards supporting the PXE protocol.

3.2. Basic logical level

Here, we refer to the operating system and network configuration. The objective is to set up the basic functionality of each node, as well as the communication among them. In this context, the IP address, or DHCP server, for the outworld network card must be provided by your organization system manager. The second card is connected to a private network among the nodes. For building a private network, there are tree ranges of private IP addresses we can resort to, corresponding to classes A, B and C networks (for a detailed description see [8]). In particular, the very used class C range corresponds to the IP addresses interval 192.168.0.0. to 192.168.255.255 with netmask 255.255.255.0.

When beginning the operating system installation in the front-end (usually Linux or Linux-based), it is necessary to establish the partitioning of the hard disk. A simple distribution is a root partition (/), a swap partition, and an additional partition (usually called /export) to be shared among the nodes. This last partition is configured as a network file system (NFS), and it acts as a communication medium between the nodes. In fact, it is here where all the software applications and user accounts will physically reside.

After configuration of the front-end, it is necessary to configure the rest of nodes. The easiest way to install the operating system on the nodes is to perform an NFS installation. Alternatively, a LAN installation through PXE

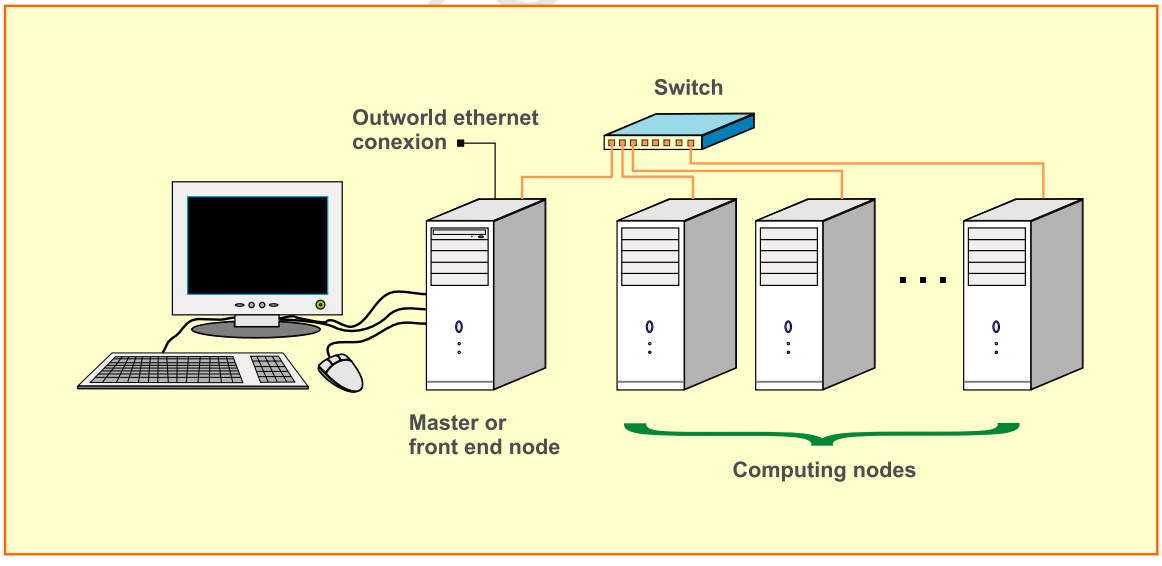

Fig. 2. Physical structure of a computer cluster.

network cards can be done. This last possibility is available, for instance, in the Rocks clustering-system distribution.

Finally, we must set up some way of communication between the machines. Thus, telnet, rsh, ssh, or rexec must be installed in all the nodes. To such an end, the appropriate packages must be selected when installing the operating system. For instance, for using the Gamess electronic structure package on a cluster, rsh must be installed. After installation, we must permanently activate the needed services. In addition, we must configure in each node, the /etc./hosts file, which contains the Internet protocol (IP) names and addresses for the local node and the other nodes in the network. This file is used to resolve a name into its IP address. In this form, any node of the system can be referred by name from any other node.

3.3. Metasystem management level

At this level, we focus on the tasks and tools related to the working of the overall system. One of the main overall tasks is the analysis of the performance of the system. Thus, we need to measure parameters like CPU load, use of RAM memory, network traffic, etc.... This information is extremely important in order to determine the actual performance, to derive performance models, and also for planning future enhancements. To register, along time, the different system parameters, we have several options. A simple one resorts to the use of Ganglia [9]. Ganglia is an open-source monitoring system for high performance computing systems, such as computer clusters and grids. Ganglia relies on two daemons. The first one (gmond), runs on each cluster node we want to monitor. This deamon is responsible for monitoring its host behavior and transmitting the information when requested. The second deamon (gmetad), when working on the cluster, is in charge of collecting the information provided by gmond about the nodes. This information is saved in round-robin databases. The information can be accessed and visualized, in real time, with any web browser, through a web front-end provided.

3.4. User and applications level

If we are building a cluster, the goal is to apply it in some manner. Therefore, it is always necessary to make some customization of the system, in order to permit the efficient use of the software tools of interest. In the present case, we are interested in quantum mechanical electronic structure calculations. As indicated above, the software to carry out the computations and tests will be Gamess, a general electronic structure package [2]. This program uses *rsh* connections to recur in several nodes of the cluster. Therefore, as indicated previously, rsh must be installed and active in the cluster. The Gamess distribution indicates how to compile the source code and configure the system in order to have an operative executable program. The reader is referred to the Gamess documentation for details.

3.5. Managing jobs

This problem is not a trivial one. Once we have our software installed and able to work in parallel, it is necessary to indicate which are the nodes where a given program must be executed. For computer clusters, there are several job managers (or queuing management systems) freely available, such as OpenPBS, SGE, or simpler systems such as GEXEC [10]. When a job is submitted through one of these job managers, a list of available nodes is generated, and used to select one (or more for a parallel case) to run the job.

4. Optimizing the cluster performance

4.1. Load measures

As a realistic test case, we consider the protonated (bioactive) form of the DBO-83 nicotinic analgesic, which is shown in Fig. 3. DBO-83 belongs to a new family of analgesics, acting on the nicotinic receptor, which looks very promising in the treatment of medium to severe pain situations [11]. To our knowledge, no detailed computational treatment of DBO-83 has been yet reported. In the present case, the molecule will be described quantum mechanically at the B3LYP/cc-pVDZ level of theory.

The use of cluster time for a full geometry optimization of DBO-83, as a function of the number of nodes considered, is monitored by ganglia. The system used was an eight computing nodes cluster. Each node has a 2.4 GHz Pentium IV processor, 1 GB SDRAM, and a 60 GB ATA/100 hard disk. Interestingly, in all of our checks, the transfer network time at the level of theory employed was found negligible (a maximum of 6 s for the worst case). For the example considered, Fig. 4 collects the disk input–output, CPU and total computation time measures.

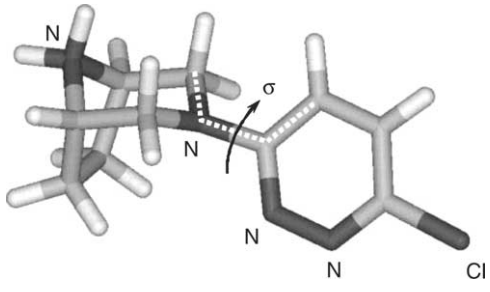

Fig. 3. Molecular structure of the protonated DBO-83 nicotinic analgesic. Carbons are represented in grey, hydrogens in white, and heteroatoms in dark grey. The figure shows the torsional angle between the two rings, with the involved atoms joined by a dashed line (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article).

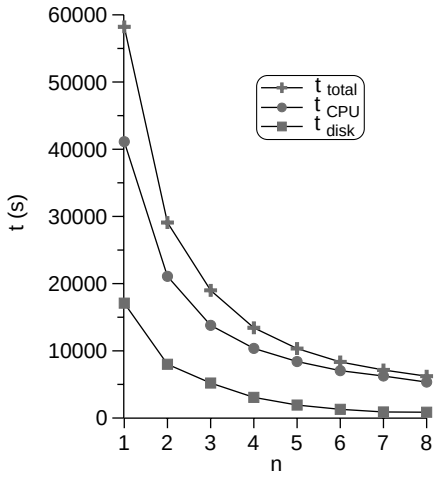

Fig. 4. Variation of CPU, disk I/O and total time (in seconds) for the complete geometry optimization of DBO-83 as a function of the number of nodes (n).

This kind of curves shows that the time, t, depends inversely on the number of nodes, n. In the simplest case, we have an inverse proportionality; doubling the number of nodes reduces the time in half. Generalizing this observation, we can propose a relationship of the form:

$$t = an^{-b} \tag{1}$$

Here, the a parameter represents the time for the one-processor case, whereas the b exponent defines the slope variation (in the simplest case b, equals 1). Despite that in a real case we are more or less far from ideal behavior, we can try functional forms similar to Eq. (1). Therefore, we can build performance models applying a regression analysis to such a functional form. Table 1 collect the results for the three cases considered in Fig. 4. It is remarkable the high value of the correlation coefficient for the three curves. Thus, the data fit properly the functional form proposed. Also we observe that the b exponent values vary from 0.9794 (less than 1.0) to 1.5163 (greater than 1.0).

These data express an interesting behavior. First, the disk input–output time (which exhibits the higher b exponent) is much more affected by the increase of nodes than the CPU time. As a consequence, when the number of nodes increases, the CPU time converges to the total time. This is a consequence of dividing the large two-electron integral file in several shorter files, which are accessed simultaneously thanks to the use of the local scratch space in each node. This effect does not depend on the algorithm (level of theory) used. In the present case (GAMESS, B3LYP), the net effect is that, for the number of nodes considered, the total time decreases almost proportionally to the number of nodes. In practical terms: it is more interesting to increase the number of nodes than to use the same number of nodes with a modest improvement of the CPU. However, in cases where the I/O gain represents a smaller proportion of the total time, the b exponent for the total time curve will be smaller than one. Thus, the gain in total time with the number of nodes will be not as good as in the present case.

Table 1
Fits to Eq. (1) of the data presented in Fig. 4

	a	b	R^2
t_{total}	60332.09	1.0908	0.9989
t_{CPU}	40979.71	0.9794	0.9996
t_{IO}	21215.10	1.5163	0.9750

The regression coefficient, R^2, is included.

Another point to highlight in the curves considered, derives from the decreasing value of the slope when increasing the number of nodes. We can analyze this question, in a general way, by using relative computing time. This magnitude is defined as the ratio between the time for n processors respect to the time for one processor. Thus, we simulate the behavior of a system following Eq. (1), with b exponent values of 0.5, 1.0, and 2.0, see Fig. 5.

We can extract some useful information from these curves. For instance, to reduce the one-processor computer time to a 10% we need a total of 100, 10 and 3 nodes, for the $b=0.5$, 1 and 2 cases, respectively. From another standpoint, considering that for typical switches in the market 24 ports are available, we evaluate the gain in time for a total of 24 nodes. The result is a reduction to a 20.41, 4.17, and 0.17% of the one-processor time for $b=0.5$, 1.0, and 2.0, respectively. For instance, in the case illustrated in Fig. 4 the one processor time is about 60,000 s. The performance model built for this case tells us that the b exponent for the total time is close to 1.0. Therefore, we can consider that 24 nodes reduce the time to 2502 s (4.17% of the initial). With these data, it is

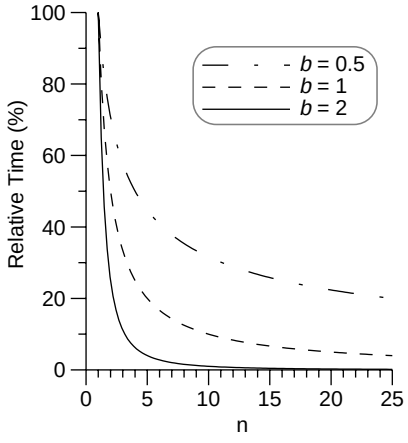

Fig. 5. Relative computing time as a function of the number of nodes (n) in the cluster. Three different variation laws are considered.

interesting to ponder if the additional (small) gain in time obtained with additional nodes is worth the increasing cost of the system.

4.2. Load balance

Another interesting issue for the use of a computer cluster, is the optimal distribution of nodes for a set of processes. This issue is somewhat different from the job scheduling problem, where the objective is to optimize the use of the whole system for a set of jobs with fixed characteristics (such as number of nodes). In our case, some good conclusions can be reached by applying some ideas borrowed from statistical thermodynamics. Fig. 4 shows the asymptotic behavior of the total computer time. Thus, for a given calculation, i, using n_i nodes, the time used, t_i, is

$$t_i = a_i n_i^{-b_i} \tag{2}$$

where a_i represents the total time used by calculation i in one computer. For processes of the same kind, a_i is different, but the b exponent is the same. Thus, considering N nodes and M processes, and a distribution of n_i nodes for each process we have

$$t_{\text{total}} = \sum_i^M a_i n_i^{-b} \tag{3}$$

In Eq. (3), t_{total} represents the cumulative time used by all the jobs. A given set of n_i values defines a job distribution. The idea is to determine the distribution that minimizes Eq. (3) subject to the constrain that the number of nodes in the cluster is fixed:

$$N = \sum_i^M n_i \tag{4}$$

To reach our goal, we can make use of a Lagrange multiplier, α. Thus, the expression to minimize becomes

$$\sum_i^M (-b a_i n_i^{-(b+1)} + \alpha)\, \mathrm{d}n_i = 0 \tag{5}$$

Then

$$n_i = \left[\frac{b a_i}{\alpha}\right]^{1/(b+1)} \quad \text{with} \quad \alpha = b\left[\sum_j^M a_j^{1/(b+1)}\right]^{(b+1)} / N^{(b+1)} \tag{6}$$

The summation in Eq. (6) represents a distribution function, Q, formally analogous to the partition function in statistical thermodynamics:

$$Q = \sum_j^M a_j^{1/(b+1)} \tag{7}$$

Therefore, using Eqs. (4), (6), and (7), we obtain,

$$n_i = a_i^{1/(b+1)} \frac{N}{Q} \tag{8}$$

Eq. (8) indicates that the overall time is minimized by assigning more computing nodes to longer jobs. It is also interesting to note, that n_i does not depend on the time each job takes on one processor, but on its $(b+1)$ root. So, if $b=1$ and we have two jobs a_1 and a_2, such that $a_1=4a_2$, the optimal distribution is to assign only $2a_2^{1/2}$ as many nodes to a_1 as to a_2.

A limiting case is found when the calculations are similar. This is the case, for instance, when performing a conformational analysis of a given molecule. Here, we have a set of independent calculations corresponding to a grid of values on the conformational coordinates. In this case, we can consider all the a_i's approximately equal and Eq. (8) reduces to

$$n_i = \frac{N}{M} \tag{9}$$

Thus, the processes can be distributed in any symmetric form. It is important to realize that this result is the same, independently of the b value.

5. Treatment of DBO-83

Applying the previous conclusion to the conformational analysis of our molecule, we prepare a total of 12 calculations for a complete span of the θ torsional angle in increments of 30°. At each θ value the rest of the geometry is fully optimized using Gamess. The number of jobs at each of the eight nodes was: 2, 2, 2, 2, 1, 1, 1, 1. In this form, we obtain the 12 results in the time it takes to have two in one computer.

Fig. 6 shows the result of our calculations as an energy diagram. The calculations evidence the existence of two

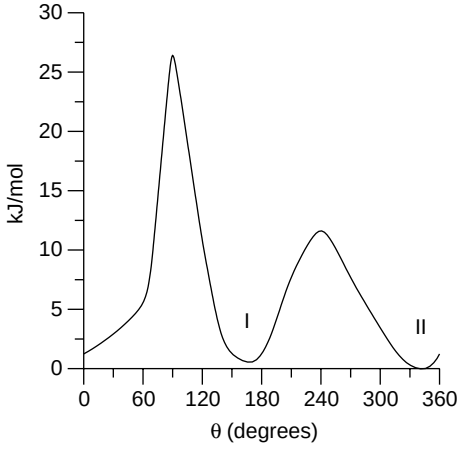

Fig. 6. Potential energy variation in DBO-83 for the intramolecular rotation of the two rings. Energy values in kJ/mol. Data obtained at the B3LYP/cc-pVDZ level.

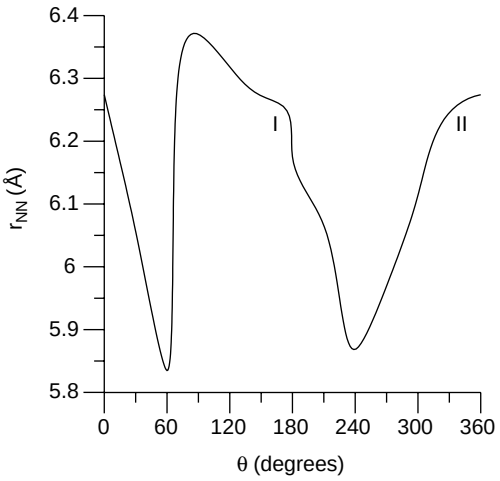

Fig. 7. Internitrogen distance, r_{NN} (Å), as a function of the θ dihedral angle in DBO-83. Conformers I and II are indicated.

energetically stable structures (minima I and II), for θ values of 171.8 and 345.2°, respectively. Minimum *I* is placed 0.6 kJ mol^{-1} above the global minimum.

These results show that DBO-83 has only two stable structures, which represent putative conformations for interaction with the receptor site in the nicotinic receptor, and the triggering of the analgesic effect.

The distance between the two nitrogens, r_{NN}, in protonated nicotine is of interest, since it is one factor defining the 3D nicotinic pharmacophore [12]. Different values have been given for r_{NN} in several studies using different nicotinic agonists [12]. All the data are in the range 4.8 Å [12a] to 6.1 Å [12d]. In DBO-83 the nitrogens corresponding to r_{NN} are the protonated nitrogen in the aliphatic moiety and the nitrogen closest to the chlorine atom in the aromatic ring, see Fig. 3. Fig. 7 shows the variation of the r_{NN} distance with the conformation. It is observed a maximum variation of 0.5 Å, corresponding to the difference between the structures for $\theta = 60$ and 90°. This variation can be compared with the 1.5 Å found in the nicotinic analgesic ABT-594 [13]. For DBO-83, the deviation from the 6.1 Å upper limit is only 0.3 Å. The conformations under 6.1 Å correspond to the ranges 30–60° and 210–270°, see Fig. 7. The first range is close to minimum II. The second conformational range corresponds to the lower maximum, between minima I and II, see Fig. 6. Thus, it seems that the conformational space close to minimum II can be a candidate for the interaction with the nicotinic receptor.

6. Conclusions

This work presents clustering computing as a useful tool in the computational chemistry field. In this context, an organization model in four abstraction levels, is presented for the building and use of a computer cluster. In order to optimize the cluster capabilities, we show how to build performance models from experimental computer load data, using electronic structure calculations. Our results show that a key factor in the performance of parallel computation in electronic structure problems is the fragmentation of the two-electron integrals file in several smaller files, local to each computing node, and asynchronously available. For the case considered (GAMESS, B3LYP), the result is a proportionality between the time of the job and the inverse of the number of computing nodes used. Also, we develop a model for the optimal distribution of computing nodes to assign to a set of given jobs. A distribution function formally analogous to the partition function is introduced. The previous conclusions are applied to the conformational and structural analysis of the new nicotinic analgesic DBO-83. In this way, two close minima are identified, separated by only 0.6 kJ mol^{-1}. By comparing the variation of the r_{NN} distance with the conformation, we find a structural similarity between the molecular arrangement close to the global minimum and previous nicotinic agonists. Thus, this zone can be considered as a putative candidate for the bioactive conformation.

Acknowledgements

This work has been supported by the Universidad de Castilla-La Mancha, and the Junta de Comunidades de Castilla-La Mancha (grant # PAI-02-001).

References

[1] W.W. Hargrove, F.M. Hoffman, T. Sterling, Sci. Am. August (2001) 62.
[2] (a) www.msg.ameslab.gov/gamess/gamess.html.
(b) M.W. Schmidt, K.K. Baldridge, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S.J. Su, T.L. Windus, M. Dupuis, J.A. Montgomery, J. Comput. Chem. 14 (1993) 1347.
[3] www.emsl.pnl.gov/docs/nwchem/nwchem.html.
[4] dasher.wustl.edu/tinker/.
[5] www.beowulf.org/showcase/index.html.
[6] www.linuxhpc.org: 'Reference' section, 'Software' link.
[7] www.rocksclusters.org/Rocks/.
[8] (a) T.L. Sterling, J. Salmon, D.J. Becker, D.F. Savarese, How to Build a Beowulf, MIT Press, Cambridge, MA, 1999.pp. 106–107;
(b) R.G. Brown, Engineering a Beowulf-Style Compute Cluster. Online publication. Access through: www.beowulf.org/overview/howto.html.
[9] ganglia.sourceforge.net.
[10] (a) OpenPBS: www.openpbs.org.
(b) SGE: gridengine.sunsource.net/.(c) GEXEC: www.theether.org/gexec/.

[11] (a) D. Barlocco, G. Cignarella, D. Tondi, P. Vianello, S. Villa, A. Bartolini, C. Ghelardini, N. Galeotti, D.J. Anderson, T.A. Kuntzweiler, D. Colombo, L. Toma, J. Med. Chem. 41 (1998) 674–681;
(b) M.W. Decker, L.E. Rueter, R. Bitner, Curr. Top. Med. Chem. 4 (2004) 369.
[12] (a) W.H. Beers, B.L. Earl, Nature 228 (1970) 917;
(b) R.P. Sheridan, R. Nilakantan, J.S. Dixon, R. Venkataraghavan, J. Med. Chem. 29 (1986) 899;
(c) R.A. Glennon, J.L. Herndon, M. Dukat, Med. Chem. Res. 4 (1994) 461;
(d) M.A. Abreo, N.H. Lin, D.S. Garvey, D.E. Gunn, A.M. Hettinger, J.T. Wasicak, P.A. Pavlik, Y.C. Martin, D.L. DonnellyRoberts, D.J. Anderson, J.P. Sullivan, M. Williams, S.P. Americ, M.W. Holladay, J. Med. Chem. 39 (1996) 817.
[13] M. Mora, C. Muñoz-Caro, A. Niño, J. Comp.-Aided Mol. Des. 17 (2003) 713.