NextComp - Molecular Dynamics Application for Long-Range Interacting Systems on a Computational Grid Environment

Marcelo P. de Albuquerque[1], Alexandre Maia de Almeida[1], Leonardo Haas Peçanha Lessa[1], Márcio P. de Albuquerque[1], Nilton Alves[1], Luis Gregorio Moyano[1], Constantino Tsallis[1,2]

[1]Centro Brasileiro de Pesquisas Físicas (CBPF/MCT)
Rua Dr. Xavier Sigaud, 150, 22290-180 - Rio de Janeiro – RJ – Brasil

[2]Santa Fe Institute
1399 Hyde Park Road – Santa Fe – NM 87501 – USA

{marcelo, amaia, llessa, mpa, naj, moyano}@cbpf.br, tasllis@santafe.edu

Abstract. *This paper presents the ongoing activities of the NextComp project. This project has as main objective to investigate the validity of nonextensive statistical mechanics for systems with long-range interactions, in particular the Hamiltonian Mean Field model. This will be achieved through molecular dynamics (MD) simulations by a parallel program taking advantages from Charm++ language. Development and tests are carried out in a Linux cluster environment, being our interest focused on a grid environment in order to overcome the limiting factors of the simulations.*

1. Introduction

We are developing a parallel, object-oriented, molecular dynamics program for high performance simulation, referred to as NextComp, in order to analyze many-body infinite-ranged Hamiltonian systems. NextComp employs the message-driven execution capabilities of the Charm++ runtime system, allowing for parallel scaling on workstation clusters and grid environments [Albuquerque, 2005].

Our molecular dynamics (MD) program simulates a physical model called *Hamiltonian Mean Field* (HMF) which consists on *N* classical planar rotators, where each rotator *i* has as dynamical variables an angle θ_i and its conjugate momentum v_i and is defined by its Hamiltonian function [Antoni 1995]

$$H = \sum_{i=1}^{N} \frac{v^2_i}{2} + \frac{1}{N} \sum_{i,j=1}^{N} [1 - \cos(\theta(i) - \theta(j))] . \qquad (1)$$

The evolution of this system is governed by its Hamilton equations (derived from Equation 1) and this set of equations may be numerically integrated, thus making possible to know any physical observable at any time.

As computing and network technologies continue to improve high performance calculus, statistical physics domain takes an interest in use MD simulations on long-range interaction systems. Such simulations may easily exceed 10^7 elements having, additionally, a very high number of time steps as well as many realizations in order to achieve bounds which provide a desired level of statistical confidence.

The computing resources necessary to perform MD simulations typically diverge when calculating long-range interactions. It remains a great challenge to carry

out high performance of MD software because it involves the simulation of several small elements, where the computation related to each element depends globally on *all* other elements.

Another difficulty is that the physical limits of interest are commonly very high (more precisely, one is forced to deal with infinite limits). This is of great significance to the NextComp Project, for reasons that will be explained in more detail in Section 2, where we present the Hamiltonian Mean Field model and its relation with the formalism of nonextensive statistical mechanics. Section 3 shows the sequential computational problem and the parallelization strategy used to speedup our MD calculus. In section 4 we make a performance analysis presenting two concrete experiments. We close our paper with conclusions and a brief outlook.

2. The Physical Problem

Classical physics, and particularly statistical mechanics, studies systems formed by elements that interact through forces. Usually, this forces have a dependency with the distance between any two elements, *r*. In most cases, the force is strong when the inter-particle distance is small, and weak when the elements are far apart.

Depending on the intensity of these forces in the long distance limit (i.e., when $r \to \infty$), this interaction may be classified as *short* or *long range interaction*. More precisely, in a system with dimension *d*, and a force between elements that behaves asymptotically (i.e., when $r \to \infty$) as $r^{-\alpha}$, the system is long-range if $\alpha \leq d$ and short-range if $\alpha > d$. The limit case $\alpha = 0$ the system does not depend on the distance and it is called an infinite range system.

Examples of systems with long-range interactions [Dauxois 2002] are gravitational systems, Coulombian systems, dipolar systems, fractures, finite systems, etc. The properties of these systems still remain to be explained, even though their evident importance. The main challenge regarding these systems is the construction of a thermodynamics that may describe them correctly and may as well explain the similarities and differences with their short-range counterparts.

This is one of the main points of interest in nonextensive statistical mechanics [Tsallis 1988]. Nonextensive statistical mechanics is a formalism formulated by one of us (CT) in 1988, that generalizes the usual Boltzmann-Gibbs (BG) statistical mechanics. This formalism is based in a generalization of the conventional entropy $S = k \sum p \ln p$ that includes a parameter *q* (that generally depends on the dynamics of the problem, or on its universality class), $S_q = k\,(1 - \sum_i p_i^q)\,(q-1)^{-1}$ where $S_q \to S$ when $q \to 1$.

An example of a long-range system is the Hamiltonian Mean Field (HMF), a system formed by planar classical rotators.

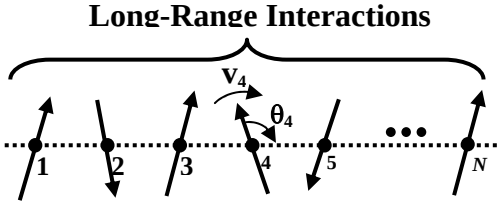


Figure 1: The Molecular dynamics uses a very large array of *N* planar rotators.

In Figure 1 we show schematically the model. The *N* rotators are symbolically depicted as black dots, and that are able to move only over the big circle. Rotators are

considered to pass through each other without collisions (which is completely equivalent to perfect elastic collisions). Each rotator *i* has an associated angle θ_i that depends in time, and therefore for any two rotators *i* and *j*, the angle between them $(\theta_i - \theta_j)$ also depends in time. Moreover, each rotator *i* has an associated angular momentum v_i (here, without loss of generality, we are assuming unitary inertial moment).

The interaction force between rotators *i* and *j* is proportional to the angle difference

$$F_{i,j} \propto sin(\theta(i) - \theta(j)) \tag{2}$$

The total force in each rotator determines its time dependence. In this model the force in each rotator is influenced by every other rotator

$$F_i = \frac{1}{2N} \sum_{j=1}^{N} sin(\theta(i) - \theta(j)) \tag{3}$$

The fact that the sum includes every rotator means that the interaction range is infinite (any rotator has an influence on any other). This simple model reflects many realistic characteristics of systems with long-range interactions such as galaxies or plasma gas [Dauxois 2002].

The thermodynamic limit of this model may be analytically solved in the microcanonical ensemble. This means that, given a specific energy (a constant in the system), the value of any mean macroscopic observable such, as the temperature, may be predicted for equilibrium. But it is known that, for certain values of initial conditions, the system may be trapped in states where the mean microscopic quantities stay approximately constant for long periods of time with different values than those predicted by the BG theory. This happens, for example, when initial conditions correspond to *waterbag* initial conditions, widely used in the literature [Anteneodo et al. 1998]. This type of initial conditions consists in $\theta_i = \theta_0 \quad \forall i = 1 \ldots N$, (i.e., all angles aligned and momenta taken arbitrarily from a uniform distribution, $v_i \in [-c, c] \quad \forall i = 1 \ldots N$, where *c* is a constant. This constant is properly defined fixing the total specific energy $E(v(i), \theta(i)) = Ne$. For the angles, without loss of generality, we choose $\theta_0 = 0$.

Using this *waterbag* initial conditions, quasistationary states appear for certain values of the specific energy *e*, slightly below *e*=0.75. In these quasistationary states, the temperature, defined as

$$T = \frac{\langle 2K \rangle}{N} = \frac{1}{N} \sum_{j=1}^{N} v(i)^2 \tag{4}$$

where $\langle \cdot \rangle$ means average between rotators, is almost constant in time and has a lower value than the one expected at equilibrium , $T_{QSS} < T_{BG}$ (see Figure 2). This result means that the time and size limits are not commutable ($N \to \infty \quad t \to \infty \neq t \to \infty \quad N \to \infty$), a main conjecture of nonextensive statistical mechanics. Furthermore, the duration t_{QSS} of the quasistationary state (defined, for example, as the time in which occurs the inflection point when the temperature relaxes to equilibrium) grows with the system size *N*, indicating that they are indeed relevant in the thermodynamical limit ($t_{QSS} \to \infty$ when $N \to \infty$). Consequently, our purpose is to simulate the evolution of this system for large

values of N to verify the applicability of nonextensive statistical mechanics to this model.

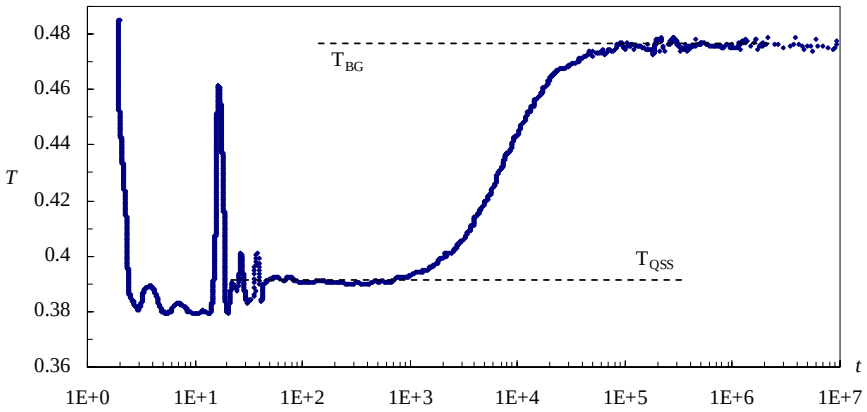


Figure 2: Temperature evolution shows a quasistationary plateau followed by a relaxation to final equilibrium.

In order to do this, we numerically simulate the Hamiltonian equations of the system. This set of differential equations define the way the rotators move

$$\left.\begin{aligned} \frac{d}{dt}\theta(i) &= v(i) \qquad\qquad 1 \le i \le N \\ \frac{d}{dt}v(i) &= m_y \cos(\theta(i)) - m_x \sin(\theta(i)) \end{aligned}\right| \tag{5}$$

where $\bar{m} = \frac{1}{N}\sum_{j=1}^{N}[\cos(\theta(j)), \sin(\theta(j))]$ is the magnetization of the system.

The total specific energy e of this system is a conserved quantity, i.e., constant at any time. To solve this set of differential equations numerically, they must be discretized, and this may have as consequence a poor total energy conservation, yielding an incorrect dynamics (i.e., the wrong time evolution). Thus, we use a special algorithm to solve sets of differential equations, the symplectic Yoshida integrator [Yoshida 1990], that guarantees a good conservation of the total specific energy and therefore correct from the physical point of view.

Additionally, the system presents statistical fluctuations in the value of its macroscopic observables (such as its temperature) due to the random character of the initial conditions (note that this is not caused by the Hamilton equations, which are deterministic). In order to reduce these fluctuations, we take averages from several realizations of the same simulation (i.e., different microscopic initial conditions consistent with equal macroscopic parameters, as for example, total specific energy).

3. NextComp Parallelization Strategy

The NextComp MD creates, as stated in Section 2, a set of N planar rotators (N>>1) where each rotator has associated a pair of state variables: its angle θ and its momentum v (see Figure 1). Numerical integration of the Hamilton equations (5) provides the state of the rotator at each time step t.

Figure 3 shows a simple diagram of the sequential NextComp MD algorithm (NextComp-MD-S). The first procedure includes initializations steps and a simulation loop (that takes account for different statistical realizations). In the simulation loop,

there is an appropriate initialization of the pair (θ, v), and preliminary iterations that represent a transient time. Then we start the integration loop calculating the dynamics.

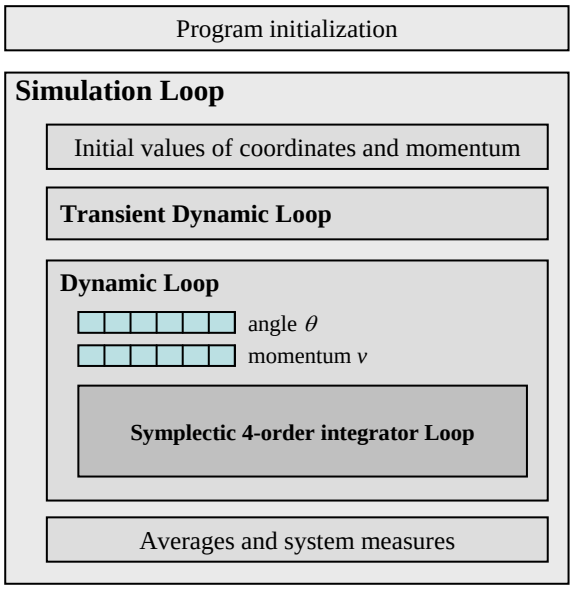


Figure 3: Diagram of the serial NextComp MD algorithm.

In this dynamical loop we use the symplectic fourth-order integration method proposed by Yoshida in order to conserve the total energy of the system for long time runs. This symplectic integration uses a four loop as follows in each time step t

1. s = 0
2. **while** s < 4
   - 2.1. $\theta(t+1,n) = f(v(t)) \ \forall N$
   - 2.2. Calculate Magnetizations $m_{x,y}(t+1) = f(\theta(t+1,n)) \ \forall N$
   - 2.3. Calculate interactions forces $F(t+1) = f(m_{x,y}(t), \theta(t+1,n)) \ \forall N$
   - 2.4. $v(t+1,n) = f(F(t+1), \theta(t+1,n)) \ \forall N$
   - 2.5. s = s + 1;
3. Calculate the kinetic energy of the system

Only after the fourth step the calculation of θ(t+1) and v(t+1) are complete. Finally some physical properties are averaged and measured.

To parallelize the NextComp MD (NextComp-MD-π) algorithm, we used the same approach as the NAMD project for Biomolecular Simulations on thousands of processors [Kale 2002]. NAMD scalability challenge was tackled adopting message-driven approaches.

NextComp dynamic components are implemented in the Charm++ parallel language [Charm 2006]. Charm++ implements an object-based message-driven execution model. In Charm++ applications, there are collections of C++ objects, which communicate by remotely invoking methods on other objects by messages. More precisely, the problem can be decomposed into objects, and since we decide the granularity of the objects, we can control the degree of parallelism. Objects encapsulate states, Charm++ objects being only allowed to directly access their own local memory. Access to other data is only possible via asynchronous method invocation directed to other objects.

Programs written in Charm++ consist in parallel objects called chares that communicate with each other through asynchronous message passing. When a chare

receives a message, the message triggers a corresponding method (or entry points –EPs) within the chare object to handle the message asynchronously. Further, chares may be organized into one or more indexed collections called chare arrays. Messages may be sent to individual chares within a chare array or to the entire chare array simultaneously.

Charm++'s parallel objects and data-driven execution adaptively overlap communication and computation, and hide communication latency: when an object is waiting for some incoming data, entry functions of other objects whose data are ready are free to execute.

For the NextComp-MD-π, we first decomposed the coupled rotators in objects where each one can only access a group of (θ, v), G. We have defined a rotator class and instantiate it as parallel chare array in a charm environment. This class was designed to use Yoshida symplectic integrator in a parallel algorithm. The first stage of the algorithm computes the angle θ and makes a reduction in $m_{x,y}$, (defined in Equation (5)), i.e. collect and sum all $m_{x,y}$ from all objects and distribute the result to all rotators objects in order to start the second stage (see Figure 4).

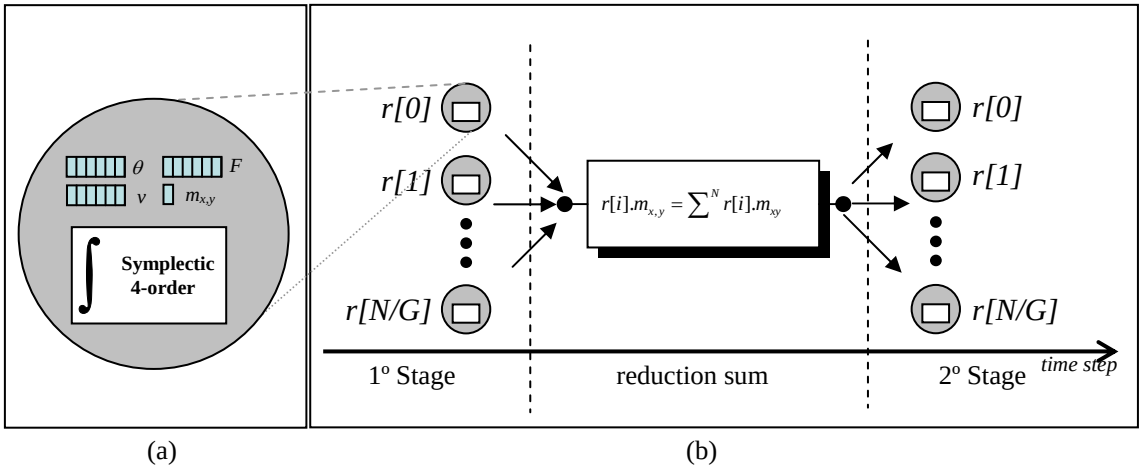


Figure 4: Diagram of the 4-order Yoshida integrator. (a) Rotator objects. (b) Parallel implementation and the synchronization stage for computation the $m_{x,y}$ sum.

The symplectic integration is the most important hotspot of the program. The parallel version needs to synchronize processes in order to compute $m_{x,y}$. NextComp-MD-π is a tightly coupled parallel program, meaning that all processes needs to exchange data at regular intervals, leading to communication latency.

The statistical mechanics point of view is focused generally in both large N and t limits. In particular, in this model (which has infinite range interactions, i.e. every pair of rotators interacts) most (common) integration algorithms will scale as $O(N^2)$, or, at most, $O(N \ln(N))$ may be achieved using optimizations taking advantage of internal symmetries [Anteneodo et al 1998]. In consequence, computational times might be unreachable, indicating that this model is a appropriate candidate for grid computing.

4. Performance Analysis

Performance is of paramount importance in parallel programming. The NextComp-MD-π development purpose should be either to solve the physical problem faster than its sequential version, and to deal with a larger physical system that could not be attained by the previously one.

We conducted three sets of experiments in the CBPF SSolar Linux Cluster [SSolar 2006] and in the NCSA[1] Xeon Linux Cluster. The first experiment accomplished was the measurement of the total execution time and thus the speedup of the system, leading to the optimal grain size computation. The second experiment corresponds to the analysis of the object execution time and the performance distribution for all object entry points. We also monitor the processors activities inspecting the program time line. Finally, the third experiment presents the method used to diminish the processors idle times.

A valuable tool to analyze in detail the parallel performance is Charm++'s *Projections*, a Java-based visualization tool (PVT) [Projections 2006]. PVT consists in an efficient automatic tracing and an interactive graphic analysis system.

All the experiments run with fixed $N=10^4$ rotators. Our choice of size N was only with the purpose of measuring performance and validates the NextComp-MD-π version. Note that N should be several orders of magnitude greater than 10^4 and the number of time steps should be large enough to verify the application of the nonextensive theory. More precisely, should be large enough to reflect the relaxation to the BG regime that, as explained in section 2, occurs at later times as N gets larger.

4.1. First Experiment

The execution time is the first metric of performance, and measures how long the NextComp-MD program runs a fixed number of time steps. Through the execution time one can compute the speedup as a function of the number of processor used in parallel calculus.

Since the heaviest computation procedure happens in the rotator-object due to the fourth-order integrator we analyzed its grain size. It was done by controlling the amount of computation per object, i.e. ranging G form 1 to *N*. In this experiment we set P= [1, 5, 10 and 20] for the number of processor and N/P as the number of rotator-objects instantiated. Table 1 presents the execution time as a function of P and G.

NextComp-MD-π — Execution Time (s)

G	P=1	P=5	P=10	P=20
1	215.50	39.92	20.59	21.90
10	38.04	8.56	6.95	8.37
20	28.27	7.54	6.22	7.77
50	23.45	6.97	5.94	7.52
100	22.73	6.90	5.82	7.28
250	21,42	6.58	5.84	7.09
500	23.51	6.81	5.64	7.13
1000	22.18	6.69	**5.62**	8.35
2000	22.45	6.43	8.29	10.69
5000	20.90	12.87	14.95	17.99
10000	20.82	23.24	25.53	29.12

Table 1: Amount of computation per object size G analysis for N=10000 rotators. The best configuration for the NextComp-MD-π is P=10 and G=1000.

[1] The NCSA proposal is related to the NextComp Project — "*Molecular Dynamics for Long-Range Interacting Systems and its Possible Connection with Non-Extensive Mechanics Theory*". NCSA Proposal Number: PHY060015.

The serial performance of the system corresponds to P=1 and G=10000, where the execution time was about 20.82 s. Table 1 shows that the best configuration of the system was achieved when G=1000 and P=10. In this condition, the speed up was ≈ 3.70 and the processor efficiency, calculated by equation (6), is $Efficiency_{10}$=37%.

$$Efficiency_P = Speedup(P) / P \quad (6)$$

Another measure of performance, similar to the efficiency, is the efficacy [Goshal 1991], useful for determining the best cost-effective number of processors for a particular application

$$Efficacy_P = Efficiency_P \; Speedup(P) \quad (7)$$

An important property of the efficacy is that its maximum corresponds to an optimal system operating point [Eager 1989]. When a new processor is added to the computation with a resulting increase in efficacy, then this gain is measured by the cost of adding it. When, on the contrary, the efficacy diminishes, then the cost of the addition outweighs the potential performance gain. For G=1000, the maximum efficacy is 1.94 for P=5. For G=2000 it is 2.10, also attained for P=5 processors (Table 2).

NextComp-MD-π Metrics

G	Speedup				Efficacy			
	P=1	P=5	P=10	P=20	P=1	P=5	P=10	P=20
1	0.10	0.52	1.01	0.95	0.01	0.05	0.10	0.05
10	0.55	2.43	2.99	2.49	0.30	1.18	0.90	0.31
20	0.74	2.76	3.35	2.68	0.54	1.52	1.12	0.36
50	0.89	2.99	3.50	2.77	0.79	1.78	1.23	0.38
100	0.92	3.02	3.58	2.86	0.84	1.82	1.28	0.41
250	0.97	3.16	3.57	2.93	0.94	2.00	1.17	0.43
500	0.89	3.06	3.69	2.92	0.78	1.87	1.36	0.43
1000	0.94	3.11	**3.70**	2.49	0.88	1.94	1.37	0.31
2000	0.93	3.24	2.51	1.95	0.86	2.10	0.63	0.19
5000	1.00	1.62	1.39	1.16	0.99	0.52	0.19	0.07
10000	1.00	0.90	0.82	0.71	1.00	0.16	0.07	0.03

Table 2: Speedups and efficacy for several configurations of the system.

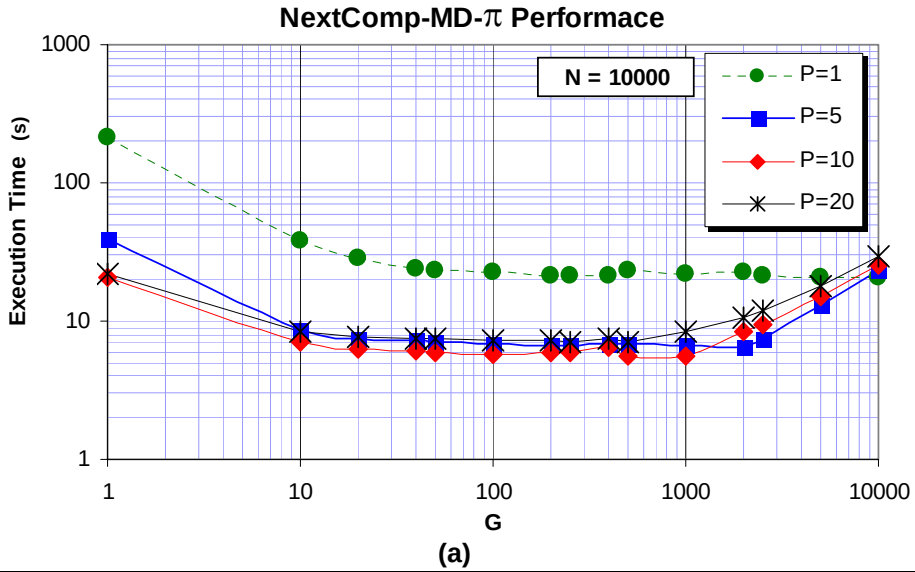
Figure 5 highlights the behavior of the execution time and the amount of computation per rotator-object (G) for NextComp-MD-π. In (a) we show that for each number of processors used in the simulations there is a minimum G, i.e. the best configuration of the system for each P. In Figure 4 (b) we put in evidence that when we add a processor to the calculation there is also a better configuration. This is expected due to the amount of time spent in communication and due to the increase of related activities with growing number of objects.

4.1. Second Experiment

We perform analysis of the object execution times for different number of entry points (EP's). Most object execution times are 54.56± 4.91 μs corresponding to the EP that computes the fourth-order integrator.

The timelines in Figure 6 show a detailed view of a NextComp-MD-π realization. It illustrates the parallel task distribution over ten processors during the runtime of the program between 5.25 s and 5.30 s. We can observe the simultaneous use

of all processors. However, the timelines graph also points out that the parallel tasks have sections of idle time. This idle time is due to the frequent periods of communication because of the synchronization events of the fourth-order integrator.



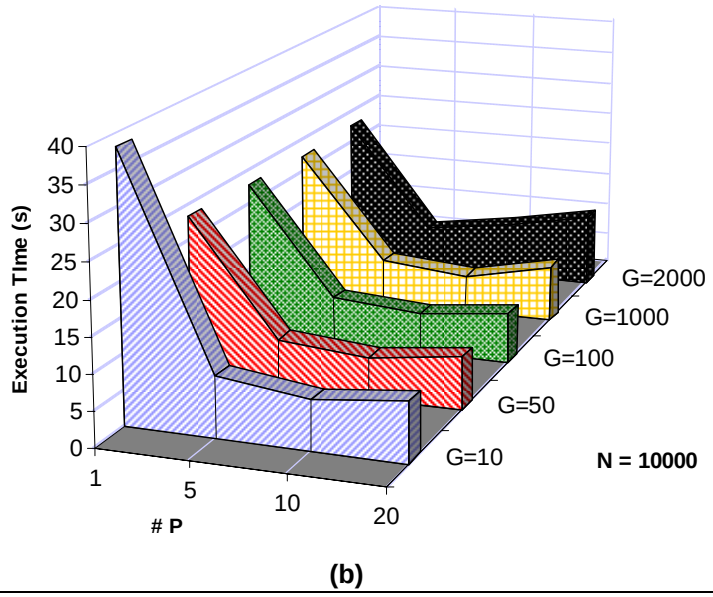


Figure 5: (a) Dependence of the execution time on the amount of computation per rotator-object G. (b) Execution time as a function of number of processors P. Note the lose of performance due to task communication when P goes from #10 to #20.

The reduction made at every time step to compute the magnetization $m_{x,y}$ produces this dependence of data synchronization. This idle time is possibly associated to the load imbalance and can affect the execution time of NextComp-MD-π.

4.3. Third experiment

In fact, to estimate the parameters of physical interest in our problem we need to generate and analyze several realizations (i.e., random samples) and obtain a statistical average of the results. These realizations are controlled by the simulation loop explained in section 3 (Figure 3). We measured the execution time for four simulations loops of NextComp-MD-π as a function of P and G. The results are shown in Figure 7. This is a

very interesting situation due to the emergence of independent tasks for each simulation. It allows Charm++ runtime system to enhance task distribution and use processors idle time. The best configuration achieved of the system in these circumstances corresponds to G=200 and P=10. As expected, the speedup increases up to ≈ 5.03 and processor efficiency increases to $Efficiency_{10,4} \approx 50\%$.

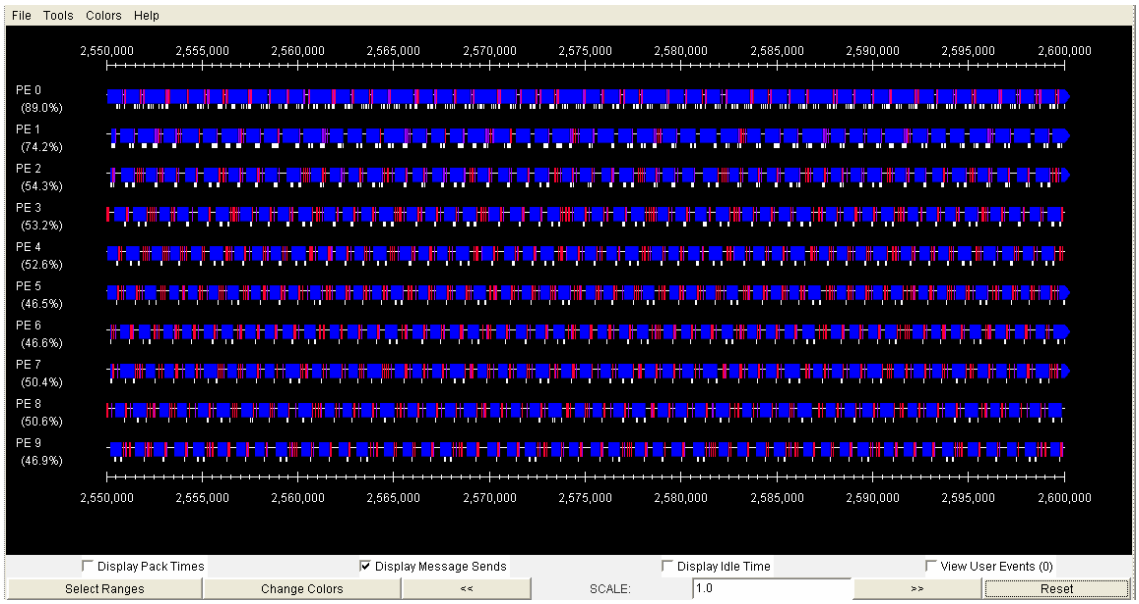


Figure 6: Charm++ Java *Projections* Timeline Tool analyzing tracedata from NextComp-MD-π. The figure presents the runtime on NCSA Xeon Linux Cluster on a window of Δ_t=40ms from 2.55 s to 2.95 s.

5. Conclusions

In this paper we presented a typical problem of computational physics that explores a system model for which no solution is known. When modeling a physical system in a computer program we have to worry about its correct mathematical representation and also with program performance. Under the statistical mechanical point of view, investigations generally focus in both large identical elements and large time limits. Computational time can be unreachable, indicating that this kind of problem is a suitable candidate for parallel computing and grid deployment.

For this reason we started the NextComp project, i.e. the development of a parallel algorithm for molecular dynamics simulation, which leads to a complex program structure. NextComp applications need to be capable of quick scalability on machines with thousands of processors and hence we decided to use message-driven objects, implemented in the Charm++ runtime system. NextComp in parallel is very challenging because of the Hamiltonian Mean Field model compels a tightly coupling application.

In this work, we performed three experiments for performance analysis. In the first one we obtained the metrics for the NextComp-MD-π; for the second one we show graphically the simultaneous processing of the program. The best parallel configuration achieved runs five times faster than its serial version. We can therefore confidently state that the results reported in this work are conservative and indicative of true performance for a production run. Nevertheless, the explored techniques show good parallel scalability allowing observations of very large physical systems.

However, we expect to achieve a better speedup by optimizations of NextComp-MD-π code. One goal is a new design of the parallel symplectic fourth-order integrator exploring the advantages of internal symmetries of the physical problem. The other one could be the use of the load balancing available in the Charm++ system that provides a variety of mechanisms and strategies to distribute tasks. Load balancing is a critical factor to achieve optimal performance in parallel applications. NextComp-MD is an iterative computation consisting in a combination of a number of time steps and calculus iterations. As the symplectic fourth-order integrator is subject to a barrier synchronization point, it forces the slowest task to determine the overall performance. Furthermore, in NextComp-MD-π the load of each individual objects is correlated with the communication patterns of consecutive iterations. In this case, Charm++ can migrate objects to create a reasonable computation and communication balance, and it also employs continuous-monitoring strategies to fine-tune load balance. Consequently, we believe that the distributing work among tasks may minimize the idle time.

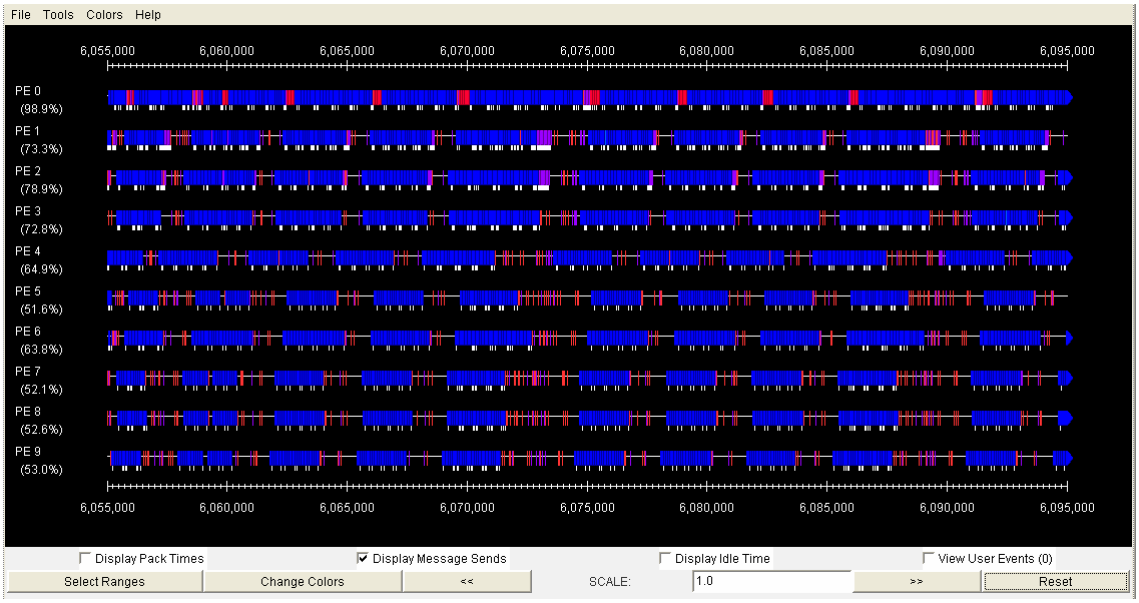


Figure 7: Timeline for NextComp-MD-π running four realizations in parallel on the NCSA Xeon Linux Cluster. Runtime on a window of Δ_t=40ms from 6.05 s to 6.09 s. The CPUs usages (left column – PE #) are higher showing a better utilization of all processors.

We intend to perform further experiments to investigate several areas with the NextComp-MD-π version described in this paper. Firstly, we intend to verify the communication latencies by measuring the time it takes to send and receive data over a HPC with dedicated high performance switches (e.g. in CBPF SSolar and NCSA Xeon Linux Clusters).

Finally, and perhaps most importantly, our future work in using parallel message-driven objects is the use of NextComp-MD-π running across multiple clusters in a Grid environment, e.g. the INTEGRIDADE Linux Cluster across the RNP Giga Network. The INTEGRIDADE Project uses Globus [Globus 2006] to create “virtual organizations” sharing computational resources owned by different real organizations. A mechanism for providing latency tolerance presents serious challenges for deployment of parallel programs in Grid environments. Masking the effects of wide-area latency is critical for achieving good performance and actually most work involving deployment of tightly-coupled applications on computational Grids has been focused on algorithm-level modifications.

6. Acknowledgment

This work was supported by the Brazilian National Council of Scientific and Technological Development (CNPq/MCT). The NextComp Project is also part of the "INTEGRIDADE - Middleware Development for Computational Grids over the infrastructure for advanced research network", of RNP (Brazilian National Education and Research Network) GIGA Project - in collaboration with National Laboratory for Scientific Computing (LNCC) and Fluminense Federal University (UFF) in Brazil and the National Center for Supercomputing Applications (NCSA) in USA.

7. References

Albuquerque, Marcelo P.; Albuquerque, M. P.; Alves, N.; Peixoto, D.; Moyano, G. L.; Tsallis. C.; Baldovin, F.; Giupponi, G. (2005); "*Dinâmica Molecular em Ambiente Computacional*", In: Annals of "III Workshop on Computational Grids and Applications".

Kale L. V., Phillips J. C., Zheng G. and Kumar S. (2002); "*NAMD: Biomolecular Simulation on Thousands of Processors*". In Proceedings of Supercomputing, The SCxy Conference series, Baltimore, Maryland, November 2002. ACM/IEEE.

Charm (2006), "*The CHARM++ Programming Language Manual – Version 5.8*" Parallel Programming Laboratory Group of the University of Illinois at Urbana-Champaign; http://charm.cs.uiuc.edu/manuals/html/charm++/manual.html, April.

Projections (2006), "*Projections Manual – Version 2.1*", Parallel Programming Laboratory at University of Illinois at Urbana-Champaign, http://charm.cs.uiuc.edu/manuals/html/projections/manual.html, April

Goshal D, Serazzi G., and Triphath S. (1991), "*The processor working set and its use in scheduling multiprocessor systems", IEEE Transactions on Software Engineering*, 17(5):443-453.

Eager D. L., Zahorjan J., and Lazowska E. D. (1989), "*Speedups versus efficiency in parallel systems", IEEE Transactions on Software Engineering*, 38:406-423.

SSolar (2006), "*CBPF SSolar Linux Cluster*", http://mesonpi.cat.cbpf.br/ssolar, April.

Globus (2006), "*The Globus Toolkit is an open source software toolkit used for building Grid system*", http://www.globus.org, April.

T. Dauxois, S. Ruffo, E. Arimondo and M. Wilkens; (2002); eds., "*Dynamics and Thermodynamics of Systems with Long Range Interactions"*, Lecture Notes in Physics Vol. 602, Springer (2002), and references therein.

C. Tsallis (1988), *J. Stat. Phys.* **52**, 479; "*Nonextensive Mechanics and Thermodynamics"*, in Non-Extensive Entropy Interdisciplinary Applications, edited by M. Gell-Mann and C. Tsallis, (Oxford University Press, Oxford, 2004).

M. Antoni and S. Ruffo (1995), *Phys. Rev. E* **52**, 2361.

C. Anteneodo and C. Tsallis (1998), Phys. Rev. Lett. 80, 5313; F. Tamarit and C. Anteneodo, *Phys. Rev. Lett.* **84**, 208 (2000); M.C. Firpo and S. Ruffo, *J. Phys. A* **34**, L511 (2001); V. Latora, A. Rapisarda, and C. Tsallis, *Phys. Rev. E* **64**, 056134 (2001);

H. Yoshida (1990), *Phys. Lett. A* **150**, 262; F. Neri (1988), "*Lie algebras and canonical integration"*, Department of Physics, University of Maryland, preprint.