

Car–Parrinello Molecular Dynamics on Massively Parallel Computers

Jürg Hutter*

*Physical Chemistry Institute, University of Zurich,
Winterthurerstrasse 190, CH-8057 Zurich, Switzerland*

Alessandro Curioni†

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

(Dated: January 31, 2005)

Abstract

The realm of applicability of ab-initio molecular dynamics simulations has continuously grown since the seminal work of Roberto Car and Michele Parrinello. Nowadays simulations on system sizes of several hundred of atoms and timescales on the order of nanoseconds are affordable. We present strategies that have been used to efficiently map the Car–Parrinello algorithm in the CPMD code to two emerging high-performance computing hardware platforms, namely, clustered shared-memory parallel servers and ultra-dense massively parallel computers, such as e.g. the IBM BlueGene/L. We show performance results and give a perspective for the future application of CPMD.

Keywords: Car–Parrinello Molecular Dynamics, Parallel Computing

*Electronic address: hutter@pci.unizh.ch

†Electronic address: cur@zurich.ibm.com

I. INTRODUCTION

Car–Parrinello (CP) molecular dynamics [1] is an efficient scheme for classical molecular dynamics (MD) where the interaction potential is defined by a variational principle. The originally proposed combination with the Kohn–Sham (KS) density functional method proved to be most successful, but the CP method has also been applied to other energy functions. [2–4] Within the KS method, the plane wave/pseudopotential framework, also applied in the original paper by Car and Parrinello, still is the most popular. However, recently progress with localized basis set methods has led researchers to reconsider the CP scheme. [5, 6]

The properties of the plane-wave basis makes it optimally suited for application with the CP scheme. Especially the orthogonality of the basis reduces the complexity of the CP equations [1, 7]. The independence of atomic positions further decouple the equations and allows an easy calculation of ionic forces using the Hellmann-Feynman theorem. This last point is of special importance for the efficiency as in the CP method ionic forces have to be calculated more frequently than in a Born-Oppenheimer MD approach. The efficiency of the CP-MD scheme (from now on we refer with this term to the combination of the KS method using plane waves with the CP method) was decisive for the application of density functional theory to new systems such as liquids or solutions. This effect was enhanced by the affinity of the plane-wave framework with the capabilities of modern computer hardware. Especially the optimal performance of CP-MD codes on parallel computers allowed the community to constantly push the limits in terms of system size and simulation time. Together with many algorithmic improvements, it is these computer code implementations of the CP-MD method that today allow routine applications to systems of few hundred atoms for tens of pico-seconds.

There are fields of application where other first principles methods are more popular, *e.g.* for chemical reactions on transition-metal surfaces or the bulk properties of metals. It is also foreseeable that in the future new methods based on linear scaling algorithms or mixed quantum mechanics/molecular mechanics (QM/MM) schemes will displace CP-MD as the method of choice for systems with several thousands of atoms. However, CP-MD is the predominant approach for liquids and solutions and has a large impact in the multiscale modeling of materials and bio-systems.

Using the CPMD code as an example, we want to show in this article that the CP-MD method provides an optimal solution for first principles simulations of such systems on future generations of supercomputers.

II. THE CPMD PROGRAM

The CPMD code is based on the original computer code written by Car and Parrinello [1]. It was been developed in the group of Michele Parrinello, first at the IBM Research Zurich laboratory and later also at the Max-Planck-Institute für Festkörperforschung in Stuttgart in collaboration with many groups world wide. It is a production code with many unique features written in Fortran 77, and has grown from its original size of approximately 10,000 lines to currently close to 200,000 lines of code. Since January 2002 the program is freely available for non commercial use [8]. Several thousand registered users in more than 50 countries have compiled and run the code on platforms as diverse as notebooks and computers at the top of the TOP500 list (www.top500.org).

The basics of the implementation of the Kohn–Sham method using a plane–wave basis set and pseudopotentials have been given in several review articles [9–12], and the CPMD code follows them closely. All standard gradient–corrected density– functionals are supported, and preliminary support for functionals that depend on the kinetic energy density is available. Pseudopotentials used in CPMD are either of the norm-conserving or the ultra-soft type [13]. Norm-conserving pseudopotentials have been the default method in CPMD, and many of the features described in the next paragraphs are currently not available for the more complicated ultra-soft scheme.

The main focus in CPMD has always been *ab initio* MD and the calculation of properties based on MD. The emphasis on MD simulations of complex structures and liquids led to the optimization of the code for large supercells and a single k-point (the $k = 0$ point) approximation. Many features have therefore only be implemented for this special case. CPMD has a rich set of features, many of them unique. For a complete overview the reader is referred to the manual [8]. The basic electronic structure method implemented uses fixed occupation numbers, either within a spin-restricted or an unrestricted scheme. For systems with variable occupation number (small gap systems and metals) the free energy functional [14] can be used together with iterative diagonalization methods.

Periodic boundary conditions are naturally implemented together with a plane wave basis. However, for special types of systems such as surfaces or charged molecules, reduced periodicity in one, two or three dimensions might be of advantage over a supercell approach. CPMD allows the decoupling periodic images in the solution of Poisson’s equation by employing adapted Green’s functions [9]. Using the velocity Verlet integrator, MD simulations can be in the NVE, NVT, or NPT ensemble. Temperature control is done through Nosé–Hoover thermostats [7] and can be applied to both ions and electrons in a CP simulation. With the help of a variety of geometrical constraints or restraints (e.g. bond, angle, torsion), thermodynamic integration or umbrella sampling can be used to calculate free-energy differences [15].

The recently developed metadynamics (mtd) method [16] provides the means to explore free energy surfaces efficiently. It is especially useful in situations where energy barriers are too high to be overcome within the time-frame of a normal *ab initio* MD simulation. Thank to its flexibility to work with a set of general collective variables, it also overcomes the restrictions of a predefined reaction path needed in thermodynamic integration.

Using the Berry phase formula [17], the dipole moment of the supercell can be calculated and used to simulate the infrared absorption spectra [18]. Maximally localized Wannier functions [19, 20] can be calculated along MD trajectories [21]. The linear response module [22] within CPMD allows the calculation of phonons (this is also possible through finite differences), NMR chemical shifts, dipole polarizabilities, Fukui functions, and several other properties. A special application of the response code is the use of a linear response basis in the $\mathbf{k} \cdot \mathbf{p}$ approximation [23], which enables an efficient calculation of the band structure of systems with weak dispersion. Also within linear response is the implementation of time-dependent density-functional theory. This module calculates optical excitation energies as well as properties and forces for excited states [24].

The inclusion of quantum effects of the nuclei is possible using the Feynman path integral method [25]. With the help of additional programs, CPMD can be used to perform QM/MM calculations [26, 27] and transition path-sampling [28].

III. PARALLELIZATION STRATEGIES

Parallelization strategies have to fulfill different requirements. Optimal data distribution to avoid memory bottlenecks usually requires additional data communication. Serial execution of certain tasks can lead to a reduction of communication but limits the maximal speedup that can be achieved.

Various strategies were followed in parallel implementations of plane-wave /pseudopotential codes [9, 29–35]. Most of them used a static work distribution, the exception being a recent implementation based on the Charm++ communication library [33]. Parallelization of the CPMD code is achieved on several levels. The central parallelization is based on a distributed-memory coarse-grain algorithm utilizing the MPI library, which is a compromise between load balancing, memory distribution and parallel efficiency. The basic scheme achieves good performance on computers with up to about 200 CPUs, depending on system size and communication speed.

On top of the basic scheme, a fine-grained, shared-memory parallelization was implemented. The two parallelization methods are independent and can be mixed. This makes it possible to achieve good performance on distributed computers with shared memory nodes and several thousands of CPUs, as well as to extend the size of the systems that can be studied completely *ab initio* to several thousand atoms.

Another parallelization strategy is targeted at the loop over electronic states needed for the calculation of the charge density and the application of the local potential. For small- and medium- sized systems the three-dimensional Fourier transform (3dFFT) within these loops dominates the computational costs. Parallelization of the 3dFFT is either limited by load balancing if a coarse-grained approach is followed or by latency in the case of fine grain parallelization. In CPMD the parallelization of the outer loop over electronic states can be combined with the parallelization of the 3dFFT. This approach (called Taskgroups) is especially suited for massively parallel computers with balanced architectures.

Some methods implemented in CPMD allow a further level of parallelization. Methods such as path-integral molecular dynamics or linear response theory are embarrassingly parallel on the level of the energy calculation. Typically, two to 32 copies of the energy and force calculation can be run in parallel. For these methods, an efficient use of computers with tens of thousands of CPUs can be envisaged.

A. Distributed-memory parallelization

The coarse-grained, distributed-memory parallelization is driven by the distribution of wave-function coefficients for all states to all CPUs. Real-space grids are also distributed, whereas all matrices that do not include a plane-wave index are replicated (especially overlap matrices). All other arrays are only distributed if this does not cause additional communications. With this scheme, all loops over plane waves, especially the ones having an N^2M scaling, where M is the number of plane waves and N the number of atoms, states or pseudopotential projectors. This scheme explicitly requires a parallel 3dFFT. Further requirements to optimize the Fourier transforms are used to find the optimal data distribution. The 3dFFT can be seen as performing the following steps:

1. Scatter of data $C(x, y, z) \leftarrow c(\mathbf{G})$.
2. Transformations along direction x .
3. Transformations along direction y .
4. Transformations along direction z .

For a general data distribution in both spaces, each of the above steps would include communication between all processors. The data distribution in CPMD minimizes the number of communication steps while maintaining optimum load balancing in both spaces. To achieve this goal the following requirements have to be fulfilled. Each processor hosts the same number of plane waves. All plane waves with common y and z components are located on the same processor. The number of different (y, z) pairs of plane-wave components is the same on each processor. A processor hosts full planes of real-space grid points. The number of real-space planes is the same on each processor. This scheme requires only a single data communication step after the first (or before the last) 1D transform. In addition, one can make use of the sparsity of the wave-function representation still present after the first transform and only communicate nonzero elements. The various load-balancing requirements are interrelated, and an heuristic algorithm to achieve near-optimum results is used. The restriction to full-plane distributions in real space, however, introduces severe problems in the case of a large number of processors. The number of planes available is typically about 50

for small systems and 200 to 300 for large systems. This restricts the maximum number of processors that can be used efficiently.

The efficiency of the basic scheme described above is limited owing to the following problems: Global summation of overlap matrices and broadcast of matrices scale as $N_{\text{pe}} \log N_{\text{pe}}$ and will become predominant for large numbers of processors (N_{pe}). The calculation of the rotation matrix in the **SHAKE/RATTLE** [7] algorithm is not parallel and limits the maximum speedup that can be achieved. Replicated overlap matrices might become a memory bottleneck for large systems on many processors with small memory. The maximum number of grid points in a direction limits the maximum number of processors that can be used efficiently for the 3dFFT. The time required for the all-to-all communications scales as $N_{\text{pe}} * \text{Latency}$, downgrading the performance scaling in the case of communication adapters with relatively high latency.

B. Shared-memory parallelization

Shared-memory parallelization on the loop level is achieved by using OpenMP compiler directives and multi-threaded libraries. Compiler directives have been used to ensure parallelization of all longer loops (those that depend on the number of plane waves or the number of grid points in real space), and to avoid parallelization on the shorter ones. This type of parallelization is independent of the MPI parallelization and can be used alone or combined with the distributed-memory approach. Tests on various shared-memory computers have shown that efficient parallelization of up to 16 processors can be achieved. It is not surprising that loop-level parallelism is very effective in CPMD. The code has a vectorization degree of more than 99% and routinely reaches more than 75% efficiency on vector processors. The combined approach is especially interesting because the shared-memory parallelization is also effective in the serial parts of the distributed-memory scheme, e.g. the rotation matrix in the **RATTLE/SHAKE** algorithm and overlap matrixes. In addition, for a given total number of processors N_{pe} , the number of tasks involved in the distributed-memory parallelization can easily be decreased by one order of magnitude, thus drastically reducing the impact of the latency in the all-to-all communications, and obtaining good scaling behavior for up to thousands of processors or enhancing the performance on loosely coupled clusters. [36]

C. Taskgroups

In systems where the 3dFFT within the loops of electronic states dominate computational costs, the efficiency of parallelization using the distributed-memory approach described before will be limited by the load balancing for large number of processors. To overcome this limitation, a method based on processor groups has been implemented [9]. For the two most important routines in which the real-space grid load-balancing problem appears, the calculation of the charge density and the application of the local potential, a second level of parallelism is introduced. The processors are arranged into a two-dimensional grid, and groups are built according to the row and column indices. Each processor is a member of its column group and its row group. In a first step a data exchange in the column group assures that all data needed to perform Fourier transforms within the row groups are available. Then each row group performs the Fourier transforms independently, and in the end another data exchange in the column groups rebuilds the original data distribution. This scheme needs roughly double the amount of communication of the original scheme. Advantages are the improved load-balancing for the Fourier transforms and the larger data packages in the matrix transposes, which also reduce the problems related to latency in the all-to-all communication needed in the matrix transposition. Moreover, this approach is particularly useful for massively parallel computers with torus connectivity, such as the IBM BlueGene/L, if the processor groups are optimally aligned with the hardware layout.

IV. BENCHMARKS

The parallelization strategies described in the preceding section have been tested on two types of general-purpose supercomputers. The first is a clustered SMP server, which is an ideal testbed for the dual-level parallelization scheme. This system consists of 40 IBM pSeries 690 32-way servers (based on the Power4 1.3 GHz processor), logically partitioned in 160 8-way SMP nodes, connected via dual-channel colony switches (Phase I system at HPCx - Daresbury). This results in an aggregate compute power of 5.2 TFlop/s. The second supercomputer is the novel IBM BlueGene/L solution, consisting of 1024 dual-processor nodes based on the PowerPC 440 embedded processors with 700 MHz clock speed, packaged in a superdense way and connected via three diverse connection hardware, namely a 3D

torus, a global tree and a standard GB Ethernet. The aggregate power of such a BG/L rack is 5.6 TFlop/s. The taskgroup scheme is ideal suited to achieve efficient scaling on this machine.

The first system investigated is solid SiC with a supercell containing 1000 atoms (2000 Kohn-Sham states) using a cutoff of 60 Ry, norm-conserving pseudopotentials and Becke-LYP functional; this system has been chosen to emphasize the benefit of the above described dual-level parallelization scheme. In table I timings in seconds for a single MD step on the clustered SMP server are shown. The results show clearly that the dual level parallelization

TABLE I: 1000 atoms SiC on clustered p60 frames

Number of processors	MPI tasks	SMP threads	Time/step (s)
512	64	8	99.4
1024	256	4	71.9
1024	128	8	56.3
1232	154	8	52.1

scheme allows an efficient adaption of the CP-MD algorithm to clustered SMP servers. This is demonstrated by the high, about 90%, parallel efficiency when scaling from 512 to 1024 processors and by the degradation in performance when going from a 128x8 mixing scheme to a 256x4 one. The time per step of about 50 s (estimated to reduce to about 25 s per step on Phase II of the HPCx system) is comparable with standard applications of CPMD to large systems. It allows a molecular dynamics simulation to be run at a rate of approximately 2.5 ps/week.

The second system investigated is water at 1 g/ml density simulated using a 32-molecule supercell with a plane-wave cutoff of 100 Ry, norm-conserving pseudopotentials and the PBE functional. This test system has been heavily used as prototype model for liquid water in the past. Moreover, because of its comparably small size of 96 atoms, 128 KS states and a real-space mesh of dimension 128^3 , its scaling is limited to 64 processors. Scaling problems are due to the strong load imbalance and the fact that communication time is dominated by latency owing to the small communication package sizes. Results for a single MD step obtained for this system on BG/L are shown in table II. From the timings it can be seen

TABLE II: 32 water molecules at 1g/l density on BlueGene/L

Number of processors	Time/step (s)
16	6.6
32	3.35
64	1.85
128	0.97
256	0.64
512	0.45

that the taskgroup parallelization scheme combined with the optimal mapping to the BG/L communication system allows an efficient scaling to up to 512 processors even for this rather small system. The same system with a smaller cutoff of 70 Ry requires 0.35 s per MD step on 512 processors, allowing a throughput of 175 ps/week.

As final test case, we report results obtained on a complex liquid/vapor interface of methanol. The system consists of 720 atoms and a 70 Ry plane-wave cutoff was used together with the PBE functional. The computational box was an orthorhombic cell with a real-space mesh of dimensions 768x160x160. The results are reported in table III. The

TABLE III: Methanol liquid/vapor interface on BlueGene/L

Number of Processors	Time/step (s)
128	1 75.2
256	1 36.0
512	1 29.53
512	2 17.98
1024	1 14.9
1024	2 14.1
1024	4 10.6

limited amount of memory per node available on BG/L requires that a system of this size to be distributed over at least 128 processors. Also for this system, as was the case for

the water example, compute time is dominated by the 3dFFTs over the KS states. The taskgroup implementation of CPMD allows this system to be scaled up to more than 1000 processors. The achieved time per CP step of less than 11 s, allows about 6 ps/week of molecular dynamics.

V. SUMMARY

The work of Car and Parrinello [1] started the field of *ab initio* molecular dynamics. Since then the CP scheme has been applied to many different simulation methods, but the originally proposed combination with Kohn–Sham-based density-functional theory in the pseudopotential/plane-wave framework has proved to be the most successful one. Started in the realm of semi-conductor solid-state physics, its combination of accuracy and flexibility allowed the method to have a large impact in many different fields, most noticeably in liquids and solutions, catalysis and enzymatic reactions. Another non negligible reason for its success is, that the CP-MD method could be adapted optimally to the emerging parallel-computer platforms in the nineties. Combining the increase in computer power (about a factor of 300 in the past 10 years) with algorithmic improvements allowed pushing the limits of simulations to larger systems and longer time scales.

In this paper we have shown parallelization strategies that allow the enhancement of the scalability of the standard Car-Parrinello algorithm on two emerging family of supercomputers, namely, clustered SMP servers and ultra-dense supercomputers. Results on diverse physical systems having sizes ranging from 100 to 1000 atoms exhibit good scalability to thousands of processors and molecular dynamics throughputs ranging from 2 to 200 ps/week.

These results make us confident that the CP-MD method will continue to play an important role in *ab initio* molecular-dynamics simulations also in the future. Most noticeably for systems ranging up to 1000 atoms and in connection with multi-scale modeling, both for length and time scales, CP-MD will remain a leading method. It will have a continuing impact among others in materials science, simulation of liquids and biological systems.

Acknowledgments

We thank Michele Parrinello for his inspiration, guidance and support during the past twelve years. We would also like to thank all the people who contributed over the years to the CPMD code. One of us (A.C.) would like to thank Wanda Andreoni for her constant supports.

-
- [1] R. Car, M. Parrinello, Phys. Rev. Lett., **1985**, 55, 2471-2474.
 - [2] D. A. Gibson, E. A. Carter, J. Phys. Chem., **1993**, 97, 13429-13434.
 - [3] S. W. Rick, S. J. Stuart, B. J. Berne, J. Chem. Phys., **1994**, 101, 6141-6156.
 - [4] M. Sprik, M. L. Klein, J. Chem. Phys., **1988**, 89, 7556-7560.
 - [5] H. B. Schlegel, J. M. Millam, S. S. Iyengar, G. A. Voth, A. D. Daniels, G. E. Scuseria, M. J. Frisch, J. Chem. Phys. **2001**, 114, 9758-9763.
 - [6] C. Raynaud, L. Maron, J. P. Daudey, F. Jolibois, Phys. Chem. Chem. Phys., **2004**, 6, 4226-4232.
 - [7] a) M. E. Tuckerman and M. Parrinello, J. Chem. Phys., **1994**, 101, 1302-1315; b) M. E. Tuckerman and M. Parrinello J. Chem. Phys. **1994**, 101, 1316-1329; c) J. Hutter, M. E. Tuckerman, and M. Parrinello, J. Chem. Phys. **1995**, 102, 859-871.
 - [8] CPMD V3.9, Copyright IBM Corp. 1990-2003, Copyright MPI für Festkörperforschung, Stuttgart, 1997-2001. See also <http://www.cpmc.org>.
 - [9] D. Marx, J. Hutter, *Ab-initio* molecular dynamics: Theory and implementation, in: Modern Methods and Algorithms of Quantum Chemistry, J. Grotendorst (Ed.), NIC Series, Vol. 1, FZ Jülich, Germany, 2000; see also <http://www.fz-juelich.de/nic-series/Volume1>.
 - [10] D. K. Remler and P. A. Madden, Molec. Phys., **1990**, 70, 921-966.
 - [11] G. Galli and A. Pasquarello, in *Computer Simulation in Chemical Physics*, eds. M. P. Allen and D. J. Tildesley (Kluwer, Dordrecht, 1993).
 - [12] G. Galli and M. Parrinello, in *Computer Simulations in Materials Science*, eds. M. Meyer and V. Pontikis (Kluwer, Dordrecht, 1991).
 - [13] D. Vanderbilt, Phys. Rev. B, **1990**, 41, 7892-7895.
 - [14] A. Alavi, J. Kohanoff, M. Parrinello, D. Frenkel, Phys. Rev. Lett., **1994**, 73, 2599-2602.

- [15] A. Curioni , M. Sprik , W. Andreoni , M. Parrinello et al., J. Am. Chem. Soc., **1997**, 119 (31): 7218-7229.
- [16] M. Iannuzzi, A. Laio, M. Parrinello, Phys. Rev. Lett., **2003**, 90, art. no. 238302.
- [17] R. D. King-Smith, D. Vanderbilt, Phys. Rev. B, **1993**, 47, 1651-1654.
- [18] M. Bernasconi, P. L. Silvestrelli, M. Parrinello, Phys. Rev. Lett., **1998**, 81, 1235-1238.
- [19] N. Marzari, D. Vanderbilt, Phys. Rev. B, **1997**, 56, 12847-12865.
- [20] G. Berghold, C. J. Mundy, A. H. Romero, J. Hutter, M. Parrinello, Phys. Rev. B, **2000**, 61, 10040-10048.
- [21] B. Kirchner, J. Hutter, J. Chem. Phys., **2004**, 121, 5133-5142.
- [22] A. Putrino, D. Sebastiani, M. Parrinello, J. Chem. Phys., **2000**, 113 7102-7109.
- [23] M. Iannuzzi, M. Parrinello, Phys. Rev. B, **2001**, 64, art. no. 233104.
- [24] J. Hutter, J. Chem. Phys., **2003**, 118, 3928-3934.
- [25] M. E. Tuckerman, D. Marx, M. L. Klein, M. Parrinello, J. Chem. Phys., **1996**, 104, 5579-5588.
- [26] M. Eichinger, P. Tavan, J. Hutter, M. Parrinello, J. Chem. Phys., **1999**, 110, 10452-10467.
- [27] A. Laio, J. VandeVondele, U. Rothlisberger, J. Chem. Phys., **2002**, 116, 6941-6947.
- [28] P. G. Bolhuis, C. Dellago, P. L. Geissler, D. Chandler, J. Phys.; Cond. Matter, **2000**, 12, A147-A152.
- [29] K.D. Brommer, B.E. Larson, M. Needels, J.D. Joannopoulos, Computers in Physics **1993**, 7(3), 350-362.
- [30] L.J. Clarke, I. Štich, M.C. Payne, Comp. Phys. Comm. **1993**, 72, 14-28.
- [31] J. Wiggs, H. Jónsson, Comp. Phys. Comm. **1994**, 81, 1-18; Comp. Phys. Comm. **1994**, 87, 319-340.
- [32] C. Cavazzoni, G.L. Chiarotti, Comp. Phys. Comm. **1999**, 123, 56-76.
- [33] R. V. Vadali, Y. Shi, S. Kumar, L. V. Kale, M. E. Tuckerman, G. J. Martyna, J. Comp. Chem., **2004**, 25, 2006-2022.
- [34] M. E. Tuckerman, D. A. Yarne, S. O. Samuelson., A. L. Hughes, G. J. Martyna, Comp. Phys. Comm., **2000**, 128, 333-376.
- [35] R. Gruber, P. Volgers, A. De Vita, M. Stengel, T.-M. Tran, Future. Gen. Comp. Sys., **2003**, 19, 111-120.
- [36] J. Hutter and A. Curioni, Parallel Computing, (2005) in press.