A New Massively Parallel Version of CRYSTAL for Large Systems on High Performance Computing Architectures

Roberto Orlando,[a] Massimo Delle Piane,[a] Ian J. Bush,[b] Piero Ugliengo,[a] Matteo Ferrabone,[a] and Roberto Dovesi*[a]

Fully *ab initio* treatment of complex solid systems needs computational software which is able to efficiently take advantage of the growing power of high performance computing (HPC) architectures. Recent improvements in CRYSTAL, a periodic *ab initio* code that uses a Gaussian basis set, allows treatment of very large unit cells for crystalline systems on HPC architectures with high parallel efficiency in terms of running time and memory requirements. The latter is a crucial point, due to the trend toward architectures relying on a very high number of cores with associated relatively low memory availability. An exhaustive performance analysis shows that density functional calculations, based on a hybrid functional, of low-symmetry systems containing up to 100,000 atomic orbitals and 8000 atoms are feasible on the most advanced HPC architectures available to European researchers today, using thousands of processors. © 2012 Wiley Periodicals, Inc.

DOI: 10.1002/jcc.23072

Introduction

High performance computing (HPC) has become the method of choice in those areas of science and technology that require the treatment of large amounts of data or the accomplishment of particularly demanding computational tasks. HPC systems can provide dedicated architecture and environment, large storage capabilities, high modularity, and fast networks. These architectures are constantly evolving, both in performance (petaflops) and in power efficiency.[1] Their relevance affects several and constantly increasing fields, spanning chemistry, biology, physics, mathematics, engineering, and design.[2,3] Chemistry has been strongly influenced by this evolution that has made the theoretical description of large and complex chemical systems possible at a very high accuracy level, for the prediction of properties and behavior of a number of systems of chemical interest.[4] One of the areas that greatly benefits from the use of HPC architectures is materials science: surfaces and interfacial phenomena, defective solids, biomaterials, and nanoparticulate systems, all require models that are hardly handled by desktop computing architectures due to the large system size.[5]

CRYSTAL[6–8] is a periodic code that adopts a local basis set of Gaussian type orbitals. A massively parallel (MPP) version of CRYSTAL09 has been distributed for about 2 years.[6] Recent modifications reported in Ref. [9] showed good scalability up to thousands of processors. In this work, important improvements in many different directions (parallelization, memory, and scalability) are discussed that have been introduced in the code with respect to the version used for Ref. [9]. The performance of the code is tested on different HPC architectures for a complex chemical system of pharmaceutical interest as a test case, the mesoporous silica MCM-41,[10] consisting of a large primitive cell ($41 \times 41 \times 12$ Å) with 579 atoms ($H_{102}O_{335}Si_{142}$) and no point symmetry (Fig. 1). The MCM-41 molecular sieve was proposed in 2001 as a drug delivery system[11] and since then mesoporous materials have been a subject of increasing attention.[12] MCM-41 is characterized by a long range order (with an ordered network of hexagonal pores), while it is amorphous at the short range. A realistic model for this material was proposed by some of us in a previous work[10] and it is the one used in this study. Unit cells of larger size have been obtained by generating MCM-41 supercells.

It is clear that also many other interesting systems are within the size of the prototype MCM-41 studied here. For instance, ZSM-5, a microporous aluminosilicate widely used as a heterogeneous catalyst for hydrocarbon isomerization reactions in petroleum industry, has a unit cell containing 288 atoms, about a half of those present in MCM-41. This means that the quantum mechanical modeling of adsorption and reactions in this system is well within the capabilities of the present code without the need to resort to more approximate quantum mechanics (QM)/molecular mechanics (MM) techniques.[13] Applicability of MPPCRYSTAL to systems of chemical interest is not limited to crystalline systems. Due to the localized nature of the Gaussian basis functions, isolated molecules can be efficiently and rigorously treated, at variance with plane wave-based codes. For example, small proteins like the vegetal protein crambin (46 aminoacids, 893 atoms)[14] are within the reach of the present code. The benefit of being able to optimize in a full quantum-mechanical way biomacromolecules is very high, considering that almost only molecular mechanics or semiempirical approaches have been used so far for this type of problems.[15]

[a] *R. Orlando, M. Delle Piane, P. Ugliengo, M. Ferrabone, R. Dovesi
Università di Torino, Department of Chemistry and NIS (Nanostructured Interfaces and Surfaces), Centre of Excellence, via Giuria 7, 10125 Torino, Italy
E-mail: roberto.dovesi@unito.it*

[b] *I. J. Bush
The Numerical Algorithms Group (NAG), Wilkinson House, Jordan Hill Road, Oxford OX2 8DR, United Kingdom*

In the next sections, after describing the structure of the CRYSTAL code and new updates, we will try to answer the following questions:

a. Is it possible to perform total energy calculations for systems with a relatively large size (up to 5000–10,000 atoms and 100,000 atomic orbitals, AOs) with a HPC system in standard conditions (that is, using the number of cores usually allocated to a "normal" user, whatever the total number of cores and memory available on the supercomputer)?

b. What is the scaling of the wall clock time with respect to the number of cores?

c. How does memory allocation scale with the number of cores and size of the system?

e. What is the performance of CRYSTAL for different HPC architectures?

f. How many cores should be used for a MPPCRYSTAL run?

Answers to the above questions are expected to provide end users with criteria for deciding the amount of resources to be allocated for a calculation. It is also expected that technical information about scalability of the code will facilitate tailoring of in-house computing facility based on commodity PC-based clusters as a function of the system size of interest for a given research group.

More generally, it is clear that the ideal situation, when very large systems are considered, is a very good scalability with respect to both the system size and the number of cores. In this article, it will be shown that CRYSTAL in its massive parallel version is very well behaving as far as scalability with the number of cores at fixed system size is concerned. As for scalability with the system size, internal tests showed that the code is linear scaling for systems up to 200 atoms in the unit cell, but scaling becomes quadratic for larger systems, as the cases considered in this article. The reasons behind are deeply bound to how the original structure of the code was designed about 40 years ago, in which the target system was very small (up to 10 atoms in the unit cell) and a lot of effort was spent to efficiently handle quantities related to translational invariance. Obviously, as the cell size increases like for the MCM-41 case, quantities related to neighbor cells are less relevant as most of the interactions are "within the origin cell." This implies a deep restructuring of the core routines of the code compared to how it is now. A brief description of the work that is currently in progress or planned in the near future to bypass this and other limitations is provided in the final Perspective Works section.

In spite of the above limitation, this article shows that systems as large as X14 (a 1x1x14 MCM-41 supercell) can be run on three different MPP architectures, without memory overflows or any other technical problem, with central processing unit (CPU) costs that remain reasonable when a thousand-processor machine is available.

Computational Details

The structure of the crystal code

CRYSTAL[6–8] permits the study of systems periodic in one-dimension (1D) (polymers), 2D (slabs), and 3D (crystals). As a limiting case, also 0D systems can be investigated. A Gaussian type basis set is used, and both density functional theory (DFT) and Hartree–Fock (HF) approximations are available. Due to the use of a local basis set and suitable truncation of the exchange series, DFT calculations with hybrid functionals are very efficient.[16] Crystalline orbitals $\psi_i(\mathbf{r};\mathbf{k})$ are expanded as linear combinations of Bloch functions (BF), $\phi_\mu(\mathbf{r};\mathbf{k})$, that in turn are linear combinations of AOs, $\varphi_\mu(\mathbf{r})$:

$$\psi_i(\mathbf{r};\mathbf{k}) = \sum_\mu a_{\mu,i}(\mathbf{k})\phi_\mu(\mathbf{r};\mathbf{k}) \tag{1}$$

$$\phi_\mu(\mathbf{r};\mathbf{k}) = \sum_{\mathbf{g}} \varphi_\mu(\mathbf{r}-A_\mu-\mathbf{g})e^{i\,\mathbf{k}\cdot\mathbf{g}} \tag{2}$$

The $\varphi_\mu(\mathbf{r})$ functions are a linear combination of a set of n_G Hermite–Gaussian type functions G with predefined coefficients d_j and exponents α_j:

$$\varphi_\mu(\mathbf{r}-A_\mu-\mathbf{g}) = \sum_j^{n_G} d_j\, G(\alpha_j;\mathbf{r}-A_\mu-\mathbf{g}) \tag{3}$$

The expansion coefficients of the BF, $a_{\mu,i}(\mathbf{k})$, are calculated by solving the usual matrix equation:

$$F^{\mathbf{k}}A^{\mathbf{k}} = S^{\mathbf{k}}A^{\mathbf{k}}E^{\mathbf{k}} \tag{4}$$

for each reciprocal lattice vector $\mathbf{k}$($F_{\mathbf{k}}$, $A_{\mathbf{k}}$, $S_{\mathbf{k}}$, and $E_{\mathbf{k}}$ are the Fock, eigenvectors, overlap, and eigenvalues matrices between BF).

Four-center bielectronic (as well as monoelectronic) integrals are evaluated analytically in a very efficient way, by taking advantage of recursion relations among Hermite polynomials, spherical harmonics, and auxiliary functions at various levels.[17] Long-range Coulomb interactions are evaluated through Ewald-type techniques and multipolar expansions.[18–21] Roughly speaking, the main steps for obtaining the ground-state electron density and energy are common to all codes using a local basis[22–25] and can be summarized as follows (a schematic view of the flux of the self-consistent field (SCF) algorithm is provided in Fig. 2):

Given an initial (guess) density matrix represented in the atomic orbital (or "direct space") basis, $P^{\mathbf{g}}$:

1. Calculate the kinetic, Coulomb, and, if required, exact exchange contributions to the Fock matrix in the atomic orbital representation ("direct space representation"), $F^{\mathbf{g}}$. Only the symmetry irreducible wedge $F^{\mathbf{g},\mathrm{irr}}$ (including hermiticity) needs to be computed, the full $F^{\mathbf{g}}$ matrix being then obtained by application of all symmetry operators.

2. If the DFT exchange and correlation contribution to $F^{\mathbf{g}}$ is required, a numerical quadrature is adopted.

3. Fourier transform $F^{\mathbf{g}}$ to reciprocal space (or BF basis): $F^{\mathbf{k}}$.

4. At each $\mathbf{k}$ point diagonalize $F^{\mathbf{k}}$.

5. Using the eigenvalues from step 4, calculate the Fermi level, and hence m_k, the number of occupied crystalline orbitals at each $\mathbf{k}$ point.

6. Sum over the occupied eigenvectors to give $P^{\mathbf{k}}$ that is then back Fourier transformed to give the new "direct space" density matrix $P^{\mathbf{g}}$ to be used in step 1.

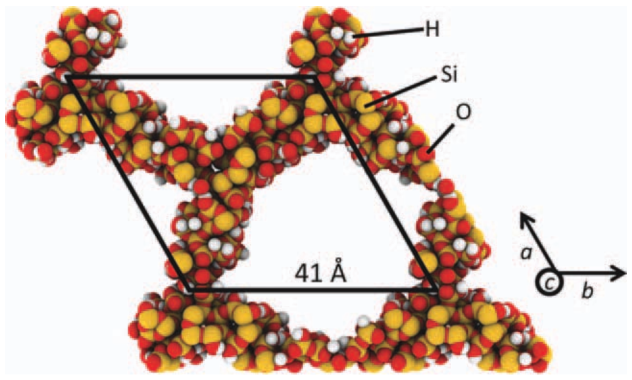

Figure 1. Unit cell of MCM-41, the X1 model system used throughout this work. Cell parameters: 41 × 41 × 12 Å. Space group: P1. Atoms per cell: 579 ($H_{102}O_{335}Si_{142}$). For tests at different system sizes, the cell was expanded (up to a factor of 14) along the *c* axis.

7. Repeat steps 1–6 until convergence in the total energy (default criterion is an energy difference between two SCF steps lower than 10^{-6} Ha).

The "direct SCF" strategy is adopted here, so as to avoid storing integrals to disk. Step 1 requires further specification. Due to the infinite nature of periodic systems, truncation criteria are adopted to limit the number of integrals to be evaluated. These criteria rely on the exponential decay of the Gaussian functions and are essentially based on screening tests: when the overlap between Gaussian functions is smaller than a given threshold, the integral is disregarded.[6,18] As an example, let us consider a typical Coulomb contribution to the Fock matrix:

$$F^{\mathbf{g}}_{\mu\nu}(\mathrm{Coul}) = \sum_{\lambda\rho} \sum_{\mathbf{g}'}^{\infty} P^{\mathbf{g}'}_{\lambda\rho} \sum_{\mathbf{h}}^{\infty} (\varphi^{\mathbf{0}}_{\mu}\ \varphi^{\mathbf{g}}_{\nu} | \varphi^{\mathbf{h}}_{\lambda}\ \varphi^{\mathbf{h}+\mathbf{g}'}_{\rho}) \tag{5}$$

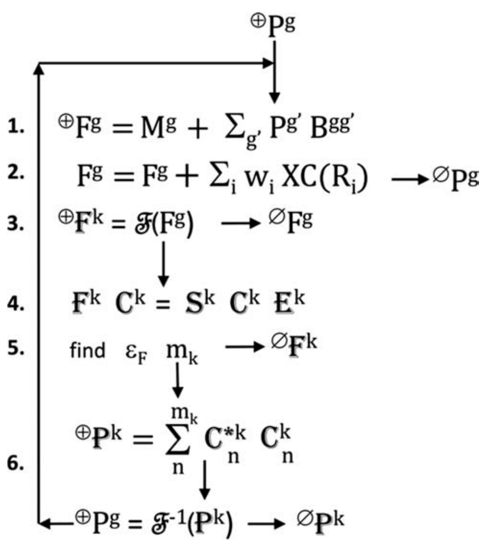
⊕Pg
1. ⊕Fg = Mg + Σg' Pg' Bgg'
2. Fg = Fg + Σi wi XC(Ri) → ∅Pg
3. ⊕Fk = 𝔉(Fg) → ∅Fg
4. Fk Ck = Sk Ck Ek
5. find εF mk → ∅Fk
6. ⊕Pk = Σn mk C*k n Ck n
⊕Pg = 𝔉-1(Pk) → ∅Pk

Figure 2. Schematic structure of the SCF algorithm employed in CRYSTAL09. Symbols in Algerian font refer to matrices which are memory distributed. Symbols ⊕/ø indicate matrices which are allocated/deallocated from memory. XC(R_i) is the exchange-correlation contribution to the Fock matrix evaluated at point R_i. See main text for the description of each numbered section.

Any integral where the overlap between $\varphi^{\mathbf{0}}_{\mu}$ and $\varphi^{\mathbf{g}}_{\nu}$ is smaller than 10^{-6} (default tolerance) is disregarded. Conversely, if the two distributions ($\varphi_{\mu}^{\mathbf{0}}$, $\varphi_{\nu}^{\mathbf{g}}$) and ($\varphi^{\mathbf{h}}_{\lambda}$, $\varphi^{\mathbf{h}+\mathbf{g}'}_{\rho}$) do not overlap, integrals are approximated by the interaction of multipole moments. Similar truncation criteria apply to the exchange series and overall five tolerances are used in CRYSTAL for the truncation of the Coulomb and exchange infinite summations. We will define these tolerances with reference to a single parameter T ($T_1 = T_2 = T_3 = T_4 = T$; $T_5 = 2 \times T$), where the tolerance itself is 10^{-T_x}, and T_x is T_1, T_2, T_3, T_4, or T_5. The size of the $P^{\mathbf{g}}$ and $F^{\mathbf{g}}$ in "direct space" is determined by T for a given fixed geometry and basis set. For exact-exchange, a larger "cluster" must be considered, and the density matrix size depends on $T_5 = 2 \times T$ (default value of T_5 is 12). For this reason, the density matrix is much larger than the Fock matrix in HF or "hybrid" DFT calculations, as shown in Table 1, where we

Table 1. Size (GB) of the main CRYSTAL matrices as a function of the system size.

Supercell	Atoms	AOs	P^g	F^g	F^k	F^k/NC	NC
X1	579	7756	0.02	0.01	0.45	0.01	32
X2	1158	15,512	0.05	0.02	1.79	0.06	32
X4	2316	31,024	0.10	0.04	7.17	0.22	32
X6	3474	46,536	0.15	0.06	16.13	0.25	64
X10	5790	77,560	0.24	0.09	44.82	0.18	256
X14	8106	108,584	0.34	0.13	87.85	0.17	512

$P^{\mathbf{g}}$ and $F^{\mathbf{g}}$ are the direct space density and Fock matrices, respectively, whereas $F^{\mathbf{k}}$ is the Fock matrix in the reciprocal space. The column labeled as NC indicates the minimum number of IBM SP6 processors (in a binary scale: 32, 64, …) on which the calculation is feasible within the 3.4 GB memory limit. Column $F^{\mathbf{k}}$/NC shows the memory occupation per core.

report the size of the main matrices for various supercells of our model system, which were obtained by expanding the cell along the *c* axis up to a factor of 14 (in the following these cells will be indicated as X1, X2…, X14). Considering one single unit, the 6-31G* Pople basis set[26] contains 7756 AOs. For a pure generalized gradient approximation (GGA) DFT calculation (such as PBE, for example), the size of $P^{\mathbf{g}}$ and $F^{\mathbf{g}}$ coincide. Table 1 also shows that $F^{\mathbf{g}}$ is much smaller than $F^{\mathbf{k}}$ (by nearly three orders of magnitude for X14) and scales linearly with the size of the system, whereas $F^{\mathbf{k}}$ increases quadratically with the number of AOs. $F^{\mathbf{g}}$ is smaller than 130 MB even for the largest system. The same applies also for $P^{\mathbf{g}}$.

A parallel algorithm has been implemented for each of the (1–6) steps, with the exception of step 5, which is not computationally demanding.

Parallelization strategies

Pcrystal. A parallel version of CRYSTAL (PCRYSTAL) based on a replicated data approach was released first in 1996:[8] all cores have access to a complete copy of all the required objects, but each core will be performing different calculations at any instant. The replication of data leads to a fairly straightforward parallelism. PCRYSTAL exploits the independence of the **k** points for the calculations in reciprocal space, that is, the Fourier and similarity transforms (Fig. 2, step 3), the diagonalization of $F_{\mathbf{k}}$ (Fig. 2, step 4) and the construction of $P_{\mathbf{g}}$ (Fig. 2, step 6). Each core is assigned a subset of **k** points and performs the calculation on that subset constructing a partial $P^{\mathbf{g}}$. The resulting code is simple and for many cases performs very well. However, it is bound to the number of **k** points (that usually decreases as the system size increases) and the amount of memory available (as all the data are replicated, the largest system that can be addressed by PCRYSTAL is no larger than that addressable by serial CRYSTAL). These limitations result in PCRYSTAL typically only scaling to a few tens of cores, with most calculations becoming impractical due to runtime before memory limits are reached.

MPPcrystal. To address the above described problems, a new massively parallel version of CRYSTAL (MPPCRYSTAL) has been developed in a previous work.[6,9] The main change from the strategy described for the original parallel version PCRYSTAL was that all large objects were distributed. In particular, in reciprocal space, the size of $F^{\mathbf{k}}$ and $A^{\mathbf{k}}$ matrices [eq. (4)] is equal to the square of the number of basis functions. MPPCRYSTAL was developed mainly to efficiently distribute the elements of these matrices among the processors and to improve load balancing when calculations with few **k** points are run on a large number of processors. Thus, all the large reciprocal space matrices were distributed and operated on in parallel. For this part, the ScaLAPACK library was used, and thus, a block cyclic distribution of the data is implemented.[27] In MPPCRYSTAL, a hierarchical parallelism is used for the reciprocal space part of the calculation; first, the independence of the **k** points is exploited and then, for each **k** point, a number of cores perform the calculation in parallel using the ScaLAPACK library.

The true novelty with respect to the MPPCRYSTAL features described in Ref. [9] is the treatment of the matrices defined in real space. In principle, the real space $F^{\mathbf{g}}$ and $P^{\mathbf{g}}$ matrices do not affect the memory requirement, as they are stored in a compact form: only nonequivalent elements (considering both symmetry and hermiticity) are included and, for each AO pair, only the significant **g** vectors are considered. For this reason, in the previous version of MPPCRYSTAL, these matrices were allocated at the beginning of the SCF cycle and deallocated at the end of a calculation. However, the memory request of the replicated $F^{\mathbf{g}}$ and $P^{\mathbf{g}}$ matrices becomes more and more demanding for systems with a large number of basis functions of the kind considered in the present case. In this version of MPPCRYSTAL, $F^{\mathbf{g}}$ and $P^{\mathbf{g}}$ are used only in their most compact "irreducible" form and are removed from memory when not in use.

Another memory impact improvement of the present version is the distribution of the matrices associated with transformations between cartesian and symmetry allowed coordinates which are essential for geometry optimization. These matrices are now generated by the iterated classical Gram–Schmidt (GS) algorithm rather than a standard modified GS method as was used in the serial version of CRYSTAL. This is because of the markedly superior parallel scaling of the former method.[28] In this way, one of the memory bottlenecks of PCRYSTAL and of the previous MPP version ($3N_a \times 3N_a$ matrices, where N_a is the number of atoms in the cell; for X14, nearly 10 GB) disappears.

A few residual serial parts in the code, which resulted to be negligible in profiling up to about 10,000 basis functions, became top CPU time consuming for larger unit cells. All these routines have been parallelized in the new version of the code.

As we will discuss in the Results section, the current major limitation of any quantum mechanical codes is the inefficient scaling provided by ScaLAPACK diagonalizer with the number of cores. This has been somewhat helped by the introduction of the faster divide and conquer routines (PDSYEVD[29] and PZHEEVD[30]) in ScaLAPACK 1.7, but it is still an issue, and this is the ultimate limitation in the scaling of the time to solution of MPPCRYSTAL.

All these improvements in the new version remove both the major limitations noted above. As a consequence the new MPPCRYSTAL version can: (i) scale to thousands of cores (test cases up to 15,000 cores have been run, showing that the code remains robust also at this limit) and (ii) address much larger problems than the previously described version of MPPCRYSTAL.[9] It is shown in the Results section that the present version of MPPCRYSTAL is able to perform SCF iterations at more than 100,000 basis functions.

Performance analysis

All the calculations run in this work shared the same fundamental computational parameters. All the runs consisted of an energy and gradient calculation performed within the density functional approximation (DFT). The Becke three-parameter hybrid exchange functional in combination with the gradient-corrected correlation functional of Lee, Yang, and Parr (B3LYP)[31–34] has been adopted for most of the calculations. Use of B3LYP implies the calculation of HF exchange. For comparison, some calculations were run adopting the Perdew, Burke, and Ernzerhof exchange-correlation functional (PBE),[35] a pure DFT functional. The Gauss–Legendre quadrature and Lebedev schemes are used to generate angular and radial points of the DFT grid, a pruned grid consisting of 75 radial points and 434 angular points, subdivided into five subintervals (LGRID). As anticipated, the values of the tolerances that control the Coulomb and exchange series in periodic systems were maintained to the default CRYSTAL values. The Hamiltonian matrix was diagonalized only at the origin of the first Brillouin zone (Γ point).[36] This choice corresponds to a small error of 0.68 mHa (1.18 μHa per atom) in the total energy for X1 if compared with the values obtained considering more **k** points (Table 2), which becomes thoroughly negligible for X2 and larger unit cells. The SCF was run in DIRECT mode, so that monoelectronic and bielectronic integrals were evaluated at each cycle. The eigenvalue level-shifting technique was used to lock the system in a nonconducting state even if the solution during the SCF cycles would normally pass through a conducting state.[6,37] The level shifter was set to 0.3 Ha. To help

Table 2. Total energy changes with respect to the reference X1 system computed with 14 **k** points, corresponding to a shrinking factor 3 along the three reciprocal lattice vectors.

Supercell	Atoms	k	ΔE (mHa)	ΔE/Atom (μHa)
X1	579	14	0	0
		8	+0.0001	+0.0002
		1	+0.681	+1.18
X2	1158	8	−0.007	−0.01
		1	−0.007	−0.01
X4	2316	8	−0.017	−0.03
		1	−0.017	−0.03

The reference energy is −66380.923805040 Ha. $\Delta E = (E_{X1,14k} - NE_{XN})/N$.

convergence of the SCF, the Fock/KS matrix at cycle *i* was mixed with 20% of the matrix at cycle *i*-1. All bielectronic Coulomb and exchange integrals were evaluated exactly (NOBIPOLA keyword),[6] as the bipolar expansion technique becomes less useful when running CRYSTAL in its massively parallel version and it is memory consuming in the present implementation. The DFT grid is distributed across the available processors (DISTGRID keyword). Matrix diagonalization is performed by the ScaLAPACK library using a divide and conquer algorithm[29] (using the DCDIAG keyword in MPPCRYSTAL), since we observed a significant speedup in this step with respect to the default diagonalization algorithm (a factor of 5 for X2).

To analyze performance and scaling of the CRYSTAL code, wall-clock time at the end of the calculation and the time required to perform some of the most important steps of the SCF were examined for an increasing number of processors (32, 64, 128, 256, 512, 1024, 2048). The scaling graphs were generated using the number of cores as an independent variable (*x*-axis), and the SPEEDUP function as the dependent variable (*y*-axis). SPEEDUP is defined as follows (a minimum number of 32 cores are used as a reference):

$$\mathrm{SPEEDUP} = \frac{T_{32}}{T_{\mathrm{NC}}} \cdot \frac{32}{\mathrm{NC}} \tag{6}$$

where T_{32} is the time obtained for the calculation on 32 cores and T_{NC} is the time obtained for the calculation on NC processors. The diagonal in each graph represents the linear scaling, that is, a scaling where doubling the number of processors results in half the time necessary to complete a task. The larger the vertical distance with respect to linear scaling, the less efficient the parallelization. Note that for calculations on X10, the minimum number of processors was 256 cores, due to memory limitations (see Table 1). Values at 32, 64, and 128 cores were extrapolated assuming a linear scaling behavior.

HPC architectures

Most of the calculations were run on the IBM SP6 system at CINECA (Italy), a P6-575 Infiniband Cluster, consisting of 5376 IBM Power6 processors (4.7 GHz), distributed across 168 computing nodes, with a theoretical peak performance of 101 TFlop/s. Each node has 128 GB of RAM available for a total system memory of 21 TB. The CINECA SP6 is provided with an Infiniband X4 DDR internal network and 1.2 PB of disk space.[38] Unless otherwise stated it should be assumed that the results presented are for this architecture.

Tests on the IBM Blue Gene P (BG/P) architecture were run on the test machine available at CINECA. Each compute card (CN) features a PowerPC 450 quad core working at a frequency of 850 MHz with 4 GB of RAM and the network connections. A total of 32 CNs are plugged into a node card. Then, 32 node cards are assembled in one rack to give a total of 4096 cores. At CINECA, only one rack is present for a peak performance of 14 TFlop/s. In BG/P systems, compute nodes are dedicated to run user application, and almost nothing else, so that an I/O node is necessary: there is one I/O node per pairs of node cards. This implies that the minimum job allocation is two node cards (64 compute cards, 256 cores).[39]

The results presented for the Cray system were run on the phase_2b component of the HECToR (UK) service. This is a Cray XE6 system and is contained in 20 cabinets and comprises a total of 464 compute blades. Each blade contains four compute nodes, each with two 12-core AMD Opteron 2.1-GHz Magny Cours processors. This amounts to a total of 44,544 cores. Each 12-core socket is coupled with a Cray Gemini routing and communications chip. Each 12-core processor shares 16 GB of memory, giving a system total of 59.4 TB. The theoretical peak performance of the phase_2b system is over 360 TFlop/s.[40]

Results

Memory requirements

A calculation for a system with a relatively large unit cell is only possible if the memory and time limitations of the computational system in use are met.

With regard to memory requirements, the unit cell of MCM-41 (7756 AOs) and a X2 supercell (15,512 AOs) can run on 32 cores. When the unit cell size increases, more and more cores are gradually required, to distribute matrices and arrays and keep the allocated memory within per node availability. Thus, a calculation of a X6 supercell (46,536 AOs) requires at least 64 processors, whereas 256 cores are the minimal requirement for running a X10 supercell that contains 77,560 basis functions. Of course, different HPC architectures are provided with variable amounts of memory per core. The industry has taken a path toward huge numbers of cores with very limited memory installed. As a consequence, memory is increasingly becoming the limiting factor in computational chemistry. The IBM BG/P architecture, for example, is provided with just 512/1024 MB (depending on the model) of memory per core. However, as most of the largest arrays in the CRYSTAL code are properly distributed, increasing the number of cores on which a calculation is run significantly reduces the occupation for each processor. For example, the X6 system, consisting of 3474 atoms and 46,536 AOs requires 1.7 GB per processor when run on 128 cores and less than 1 GB per processor when 512 CPUs are used. Thus, increasing the number of cores fits memory limits per node in most cases.

How large can a unit cell be in CRYSTAL calculations? The ideal answer would be infinite, but for the moment being, on the tested HPC architectures, the code was shown to work properly on a system consisting of 14 MCM-41 units (X14), containing 108,584 AOs. In the latter case, only few SCF cycles were run, as a complete energy calculation may become very time consuming when running on a small number of processors. However, as the runtime for one SCF step is almost constant throughout an SCF calculation it is straightforward to estimate the total SCF time.

Time versus cores

The next question is to establish the number of cores to run such calculations most efficiently. This is a critical issue, since each combination of size, symmetry, level of theory corresponds to an efficient optimal balance of running time and parallelization efficiency. In general, running a calculation of a relatively small system on a high number of cores is a waste of computational power, since only a small SPEEDUP is obtained.

To investigate this point, first a B3LYP energy and gradient calculation for the unit cell of MCM-41 (X1) was run on an increasing number of cores on the CINECA SP6 system. This corresponds to one step of a structure optimization. Wall-clock time is reported in Figure 3 (solid line), where the number of processors is correlated to the resulting SPEEDUP. This calculation, comprising 13 SCF steps and the analytic computation of the total energy gradient, takes about 5 h when performed on 32 cores that decreases to 8 min when 2048 cores are used. For this size of system, the deviation from linear scaling starts at 512 cores. For example, doubling the number of cores from 1024 to 2048 decreases the running time from 13 to 8 min, corresponding to a factor of 1.5 compared to the ideal factor of 2. Thus, parallelization efficiency (see Fig. 3 caption) for the X1 system at 2048 cores, considering the SPEEDUP from a 32 cores calculation, is only 57%.

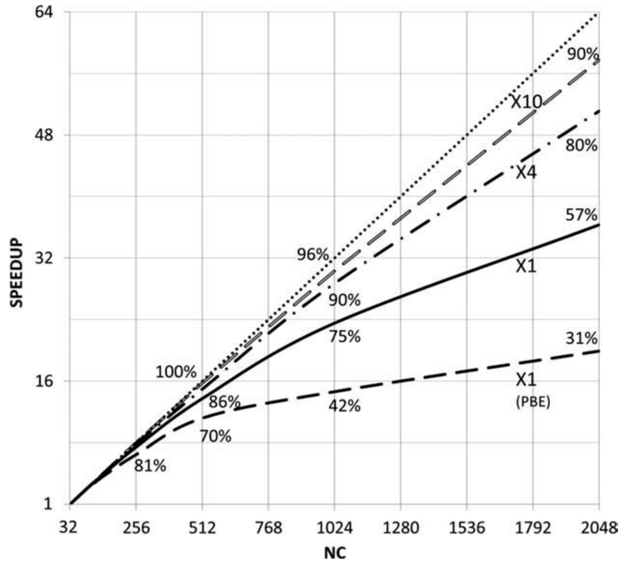

Figure 3. SPEEDUP [Eq. (6) of the text] versus the number of cores. Data are provided for SCF+Gradient calculations of a single cell (X1) of MCM-41, both at the B3LYP (solid line) and PBE (dashed line) level of theory, and for supercells of four (X4) and 10 (X10) MCM-41 units, with the B3LYP functional (dot-dashed and empty-dashed lines, respectively). The dotted line represents ideal linear scaling. For some points scaling efficiency is reported, evaluated as (SPEEDUP/NC)%.

It is expected that available computational resources are used more efficiently for large systems. This was verified by running a SCF (15 steps) and gradient computation of the MCM-41 X4 supercell (Fig. 3, dot-dashed line). The figure reveals that a significant increase in efficiency is obtained. The linear regime is maintained to as many as 1024 cores and at 2048 cores the scaling efficiency is still as large as 90%. Such a calculation requires more than 30 h when run on 64 cores but only 1.5 h if 2048 processors are used. The same kind of improvement in parallelization is observed in the case of a X10 supercell consisting of 5790 atoms and 77,560 AOs (Fig. 3, empty dashed line). With 2048 cores, a SPEEDUP of 58 with respect to 32 cores is achieved, compared to the ideal value of 64. This corresponds to a 90% parallelization efficiency.

Now, we analyze how well the different steps in a CRYSTAL calculation scale with the number of processors. Figure 4

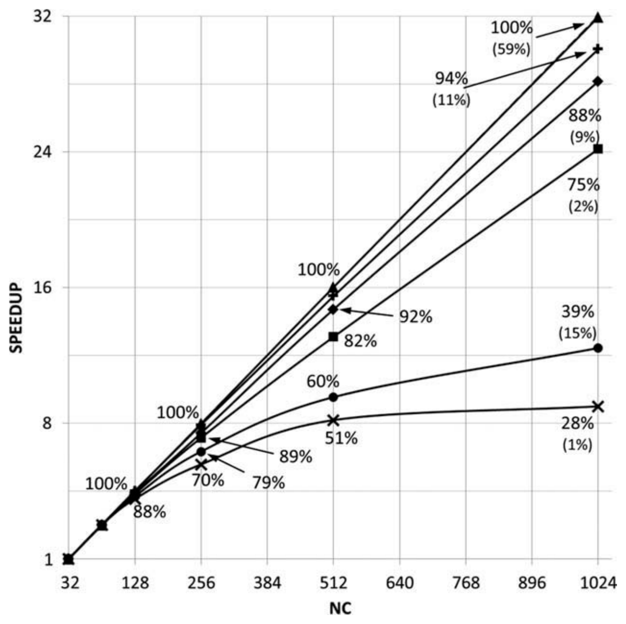

Figure 4. SPEEDUP of the main tasks in CRYSTAL09 versus the number of cores, for a B3LYP energy and gradient calculation (X4 MCM-41 unit cell): preliminary steps (×), monoelectronic (◆) and bielectronic integrals (▲) evaluation, computation of the exchange and correlation DFT contributions to the Fock matrix (F_{XC} DFT) (■), diagonalization (●), energy gradient (+). Percentage data denote efficiency. Data in parentheses report the percentage of the total running time for every task at 1024 cores. The diagonal represents the ideal linear scaling.

represents the SPEEDUP versus the number of cores in the most time consuming steps in a SCF energy calculation (see Computational Details for a general description) for a X4 supercell of MCM-41: preliminary tasks (symmetry, construction of pointers, prescreening for the monoelectronic and bielectronic integrals), integral evaluation (both for monoelectronic and bielectronic contributions), computation of the exchange and correlation DFT contributions to the Fock matrix, diagonalization, calculation of energy gradients. Diagonalization, accomplished using a Divide & Conquer algorithm,[6,29] deviates from ideal behavior at very low number of cores: when more than 256 cores are used, only a small reduction of the time requested by this routine is found. The poor efficiency of diagonalization algorithms is a known issue in linear algebra[41] and can truly become the limiting factor for the overall efficiency of a computational chemistry code.

The subroutines used in the integrals evaluation step scale with a very high efficiency. Indeed, the scaling of the bielectronic integrals computation is almost ideal, whereas the small deviation from linearity for the monoelectronic contributions can be explained by a different load balance among the processors. The linear scaling of integral evaluation routines is widely reported in the literature.[42] The computation of the XC DFT contributions scales decently. The energy gradient calculation scales in a similar way to that of the integrals.

The overall scaling of the code depends on the relative contributions of the various steps to the total running time (and contributions up to 1024 cores for the X4 supercell are reported in Fig. 4). In turn, this is linked to the level of parallelization of each part of the code (Amdahl's law).[43] In essence, the remaining serial part, or poorly parallelized section of the code, will dominate the overall efficiency in the limit of very large number of cores. Moreover, some apparently quick operations may become much more time consuming when the system size increases. For this reason, in the present version of the code, the remaining time consuming preliminary steps in a CRYSTAL calculation for very large unit cells have been parallelized. Residual serial code in this part of the calculation accounts for poor scaling at NC$>$128. However, the associated overhead is negligible ($<$1%). Computation of the bielectronic integrals is usually the most time consuming step. In our test case, their impact is slightly reduced by the amount of void in the MCM-41 model (Fig. 1).

Concerning the role of the Hamiltonian, a comparison was performed between the hybrid B3LYP and the generalized gradient-approximated PBE functionals. A PBE calculation for X1 is 2.5 times faster than the corresponding B3LYP run, because exchange series are to be evaluated only in the latter case (note that this ratio can be much larger, by up to, or more than one order of magnitude, when a PW basis set is used).[44] However, B3LYP guarantees a noticeable higher efficiency in scaling of the total time (Fig. 3), because the best scaling part of CRYSTAL, the computation of two electron integrals, accounts for 35% of the total running time with 2048 cores in the case of B3LYP and only for 12% in the corresponding PBE calculation.

Portability

In this section, we give a brief account of portability to different HPC architectures. As part of this performance analysis, the present version of the CRYSTAL09 code was run on two further machines, in addition to the IBM SP6 installation: an IBM BG/P system and the Cray XE6 supercomputer (see Computational Details for their characteristics). An SCF plus gradient calculation for the X1 MCM-41 cell was run and the total time scaling efficiency was considered (Fig. 5). CRYSTAL09 runs

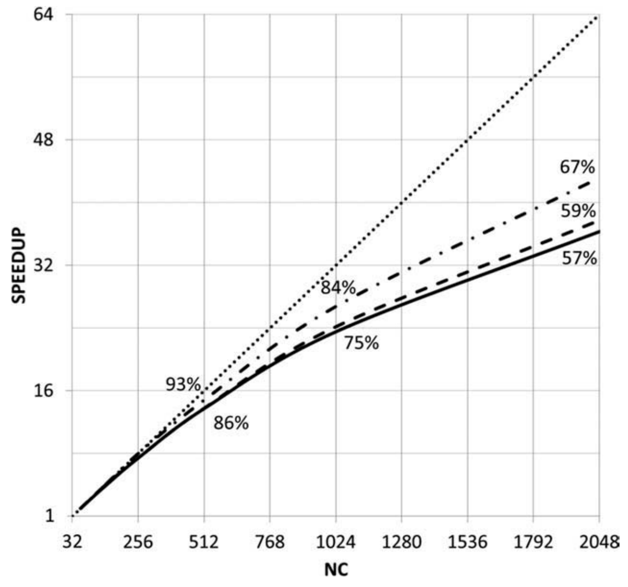

Figure 5. SPEEDUP versus the number of cores on different HPC architectures: IBM SP6 (solid line), IBM Blue Gene P (dot-dashed line), and Cray XE6 (dashed line). Data are provided for SCF+Gradient calculations on a single unit cell of MCM-41 (X1), at the B3LYP level of theory. The dotted line represents ideal linear scaling. Percentage data as in Figure 4.

properly on all the considered architectures and the computed total energy is completely machine independent (all significant digits are equal). Both BG/P and HECToR are HPC systems more focused on massive parallelization with respect to the SP6 architecture. BG/P, in particular, is strongly built on the idea of using a very high number of cores of relatively low clock frequency. These differences are mirrored in the scaling efficiency, which is highest for BG/P, lowest for SP/6 with HECToR in between, when considering the same number of processors. Nevertheless, CRYSTAL09 scaling looks very similar for all the considered architectures.

Conclusions

This article reports on the parallel performance of the CRYSTAL *ab initio* periodic program when running on HPC architectures for the study of complex materials of great importance. It is also the most complete memory usage and performance analysis of the program to date. Results of this work can help CRYSTAL users in preliminary analysis of feasibility of a calculation to assess the amount of computational resources to be allocated to run it most efficiently. The present version of the program represents a definite improvement with respect to the state of the art reported in Ref. [9] as we removed the bottlenecks appearing when the program was run on more complex systems, such as the mesoporous silica model considered here. To obtain the results reported in this article, the original code was modified following two main directions: (a) reducing memory requirements, particularly considering the constraints of modern HPC architectures and (b) speeding up new parts of the code through parallelization.

This new version of the code is capable of calculating the electronic energy of systems with up to 8000 atoms and more than 100,000 AOs on a typical HPC architecture. Such improvements can strongly enlarge the range of complex materials to be investigated, so that calculations of increasingly larger systems become feasible on a larger number of cores. Parallelization of most of the code resulted in a good to almost perfect scaling. As expected, the optimal number of cores to be used for a CRYSTAL calculation is mainly related to the size of the crystal unit cell. The performance of CRYSTAL has been checked on three rather different HPC architectures, namely the IBM/SP6, the IBM Blue Gene/P, and a Cray XE6 machines, showing extremely similar scalability. It is also reassuring that the SCF DFT total energy is machine independent with respect to the HPC architecture.

We are aware that some of our previous considerations need to be verified for different materials and that, when dealing with more complex computations (like frequency calculations), new bottlenecks may arise. For this reason, the development of the CRYSTAL code is constantly evolving and the results of further improvements in performance will be illustrated in the future.

Perspective Works

As stated in the Introduction section, the current main limitation of the CRYSTAL code remains the quadratic scaling with the system size that emerges with cells containing more than 200 irreducible atoms. To solve this problem, major interventions on the code are needed, particularly revisiting the loop structure in the main routines. However, the real difficulty is disentangling the Fock diagonalization dependence on the system size, as the time for standard diagonalization routines adopted by MPPCRYSTAL grows as O(N-AO^3). To move to a O(N-AO) dependence implies complete rethinking of the basic algorithms as the solution might be achieved by direct minimization techniques taking advantage of matrix sparsity. Unfortunately, no standard libraries are available to perform a distributed sparse matrix by matrix multiplication, a key operation in any quantum mechanical program. This last point would require a large amount of human work to reach the final target of linear scalability with respect to both the system size and the number of cores. Recently, new powerful general purpose graphical processing units (GPUs) have become available to the scientific community. They are based on a very large number of independent cores designed to satisfy high numerical intensive work behind the realistic graphical rendering of highly sophisticated computer games. It soon became clear that their power could be equivalently used to solve intensive numerical problems in computational chemistry. Regarding a possible exploitation of GPU computing by MPPCRYSTAL, one should consider that while codes based on classical force fields have been successfully ported to these new GPUs, very few quantum mechanical codes have been for the following reasons: (i) the limited amount of memory shared among one-chip GPU cores; (ii) the great deal of work needed to refurbishing complex quantum mechanical codes to suite the specificity of GPU architecture. TeraChem,[45] a full quantum mechanical code *a priori* designed to run on GPUs, is the only exception. Despite its impressive performance, the introduction of the *d* type orbitals has been completed only very recently, showing the difficulty of this step. Gaussian has announced a possible porting of Gaussian09 to CUDA as a joint project with NVIDIA and the Portland group. We are only aware of one attempt to port vienna ab initio simulation package (VASP) on GPUs[46] while Quantum ESPRESSO[47] has released a beta version only of the PWscf module on GPUs. What emerges from the above discussion is that, despite the attractive perspective to run a quantum mechanical calculation on GPUs, a lot of work is needed to port actual codes to the new architectures. This is also the case of CRYSTAL whose complexity hampers its implementation on GPUs, at least in the near future.

Acknowledgments

The authors are grateful to CINECA supercomputing Centre for supporting part of this work with the ISCRA initiative through the HP10A7WAF8, HP10AC4ZGA and HP10AN1YJ1 projects.

Keywords: HPC · DFT · B3LYP · MCM-41 · massive parallel · CRYSTAL

How to cite this article: R. Orlando, M. D. Piane, I. J. Bush, P. Ugliengo, M. Ferrabone, R. Dovesi, *J. Comput. Chem.* **2012,** *000*, 00–00. DOI: 10.1002/jcc.23072

[1] IBM Blue Gene Team, *IBM J. Res. Dev.* **2008**, *52,* 199.
[2] R. S. Saksena, B. Boghosian, L. Fazendeiro, O. A. Kenway, S. Manos, M. D. Mazzeo, S. K. Sadiq, J. L. Suter, D. Wright, P. V. Coveney, *Philos. Trans. Roy. Soc. A* **2009**, *367,* 2557.
[3] P. T. Cummings, *Fluid Phase Equilib.* **1998**, *144,* 331.
[4] A. Grama, A. Gupta, G. Karypis, V. Kumar, Introduction to Parallel Computing, 2nd ed.; Addison Wesley: Boston, (United States), **2003**.
[5] S. M. Woodley, C. R. A. Catlow, *Proc. Roy. Soc. A: Math. Phys. Eng. Sci.* **2011**, *467,* 1880.
[6] R. Dovesi, V. R. Saunders, C. Roetti, R. Orlando, C. M. Zicovich-Wilson, F. Pascale, B. Civalleri, K. Doll, N. M. Harrison, I. J. Bush, P. D'Arco, M. Llunell, CRYSTAL09, User's Manual: Torino, Italy **2009**.
[7] R. Dovesi, R. Orlando, B. Civalleri, C. Roetti, V. R. Saunders, C. M. Zicovich-Wilson, *Z. Kristallogr.* **2005**, *220,* 571.
[8] R. Dovesi, B. Civalleri, R. Orlando, C. Roetti, V. R. Saunders, *Rev. Comp. Chem.* **2005**, *21,* 1.
[9] I. J. Bush, S. Tomic, B. G. Searle, G. Mallia, C. L. Bailey, B. Montanari, L. Bernasconi, J. M. Carr, N. M. Harrison, *Proc. R. Soc. A: Math. Phys. Eng. Sci.* **2011**, *467,* 2112.
[10] P. Ugliengo, M. Sodupe, F. Musso, I. J. Bush, R. Orlando, R. Dovesi, *Adv. Mater.* **2008**, *20,* 4579.
[11] M. Vallet-Regi, A. Ramila, R. P. del Real, J. Perez-Pariente, *Chem. Math.* **2001**, *13,* 308.
[12] M. Vallet-Regi, F. Balas, D. Arcos, *Angew. Chem. Int. Ed.* **2007**, *46,* 7548.
[13] B. A. De Moor, M.-F. Reyniers, M. Sierka, J. Sauer, G. B. Marin, *J. Phys. Chem. C* **2008**, *112,* 11796.
[14] A. Schmidt, M. Teeter, E. Weckert, V. S. Lamzin, *Acta Crystallogr. F: Struct. Biol. Cryst. Commun.* **2011**, *67,* 424.
[15] J. J. P. Stewart, *J. Mol. Model.* **2009**, *15,* 765.
[16] M. D. Towler, A. Zupan, M. Causà, *Comput. Phys. Commun.* **1996**, *98,* 181.
[17] L. E. McMurchie, E. R. Davidson, *J. Comput. Phys.* **1978**, *68,* 4839.
[18] C. Pisani, R. Dovesi, C. Roetti, Hartree-Fock Ab initio Treatment of Crystalline Systems; Springer-Verlag: Berlin, Germany **1988**.

[19] R. Dovesi, C. Pisani, C. Roetti, V. R. Saunders, *Phys. Rev. B* **1983**, *28,* 5781.
[20] V. R. Saunders, C. F. Fava, R. Dovesi, C. Roetti, *Comput. Phys. Commun.* **1994**, *84,* 156.
[21] V. R. Saunders, C. F. Fava, R. Dovesi, L. Salasco, C. Roetti, *Mol. Phys.* **1992**, *77,* 629.
[22] M. F. Guest, I. J. Bush, H. J. J. Van Dam, P. Sherwood, J. M. H. Thomas, J. H. Van Lenthe, R. W. A. Havenith, J. Kendrick, *Mol. Phys.* **2005**, *103,* 719.
[23] J. M. Soler, E. Artacho, J. D. Gale, A. Garcia, J. Junquera, P. Ordejon, D. Sanchez-Portal, *J. Phys. Condens. Mater.* **2002**, *14,* 2745.
[24] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, J. A. Montgomery, J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski, D. J. Fox, GAUSSIAN09, **2009**.
[25] H. J. Werner, P. J. Knowles, R. Lindh, F. R. Manby, M. Schütz, P. Celani, T. Korona, G. Rauhut, R. D. Amos, A. Bernhardsson, A. Berning, D. L. Cooper, M. J. O. Deegan, A. J. Dobbyn, F. Eckert, C. Hampel, A. W. Lloyd, S. J. McNicholas, W. Meyer, M. E. Mura, A. Nicklass, P. Palmieri, R. Pitzer, U. Schumann, H. Stoll, A. J. Stone, R. Tarroni, T. Thorsteinsson, MOLPRO, **2006**.
[26] R. Ditchfield, W. J. Hehre, J. A. Pople, *J. Chem. Phys.* **1971**, *54,* 724.
[27] ScaLAPACK: www.netlib.org/scalapack. Accessed on June 17, 2012.
[28] F. J. Lingen, *Commun. Numer. Methods Eng.* **2000**, *16,* 57.
[29] PDSYEVD: http://www.netlib.org/scalapack/double/pdsyevd.f. Accessed on June 17, 2012.
[30] PZHEEVD: http://netlib.org/scalapack/complex16/pzheevd.f. Accessed on June 17, 2012.
[31] A. D. Becke, *J. Chem. Phys.* **1993**, *98,* 5648.
[32] C. Lee, W. Yang, R. G. Parr, *Phys. Rev. B* **1988**, *37,* 785.
[33] S. H. Vosko, L. Wilk, M. Nusair, *Can. J. Phys.* **1980**, *58,* 1200.
[34] P. J. Stephens, F. J. Devlin, C. F. Chabalowski, M. J. Frisch, *J. Phys. Chem.* **1994**, *98,* 11623.
[35] J. P. Perdew, K. Burke, M. Ernzerhof, *Phys. Rev. Lett.* **1996**, *77,* 3865.
[36] H. J. Monkhorst, J. D. Pack, *Phys. Rev. B* **1976**, *8,* 5188.
[37] V. R. Saunders, I. H. Hillier, *Int. J. Quantum Chem.* **1973**, *7,* 699.
[38] IBM SP6 (CINECA): http://www.cineca.it/en/hardware/ibm-sp65376-0. Accessed on June 17, 2012.
[39] IBM BGP (CINECA): http://www.cineca.it/en/hardware/bgp4096-0. Accessed on June 17, 2012.
[40] CRAY XE6 (HECToR): http://www.hector.ac.uk/service/hardware/phase2b.php. Accessed on June 17, 2012.
[41] R. Čiegis, D. Henty, B. Kågström, J. Žilinskas, A. G. Sunderland; Springer: New York, **2009**, pp 57–66.
[42] S. Kindermann, E. Michel, P. Otto, *J. Comput. Chem.* **1992**, *13,* 414.
[43] G. M. Amdahl, In Proceedings of the April 18–20, 1967, Spring Joint Computer Conference; ACM: Atlantic City, New Jersey, **1967**, pp 483–485.
[44] J. Paier, R. Hirschl, M. Marsman, G. Kresse, *J. Chem. Phys.* **2005**, *122,* 234102.
[45] TeraChem: http://www.petachem.com/index.html. Accessed on June 17, 2012.
[46] M. Hutchinson, M. Widom, *Comput. Phys. Comm.* **2012**, *183,* 1422.
[47] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. Fabris, G. Fratesi, S. de Gironcoli, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, R. M. Wentzcovitch, *J. Phys. Condens. Matter* **2009**, *21,* 395502.

Received: 2 February 2012
Revised: 18 June 2012
Accepted: 24 June 2012
Published online on