OPEN ACCESS Freely available online

PLOS ONE

Performance Analysis of Electronic Structure Codes on HPC Systems: A Case Study of SIESTA

Fabiano Corsetti*

CIC nanoGUNE, Donostia-San Sebastián, Spain

Abstract

We report on scaling and timing tests of the SIESTA electronic structure code for *ab initio* molecular dynamics simulations using density-functional theory. The tests are performed on six large-scale supercomputers belonging to the PRACE Tier-0 network with four different architectures: Cray XE6, IBM BlueGene/Q, BullX, and IBM iDataPlex. We employ a systematic strategy for simultaneously testing weak and strong scaling, and propose a measure which is independent of the range of number of cores on which the tests are performed to quantify strong scaling efficiency as a function of simulation size. We find an increase in efficiency with simulation size for all machines, with a qualitatively different curve depending on the supercomputer topology, and discuss the connection of this functional form with weak scaling behaviour. We also analyze the absolute timings obtained in our tests, showing the range of system sizes and cores favourable for different machines. Our results can be employed as a guide both for running SIESTA on parallel architectures, and for executing similar scaling tests of other electronic structure codes.

Citation: Corsetti F (2014) Performance Analysis of Electronic Structure Codes on HPC Systems: A Case Study of SIESTA. PLoS ONE 9(4): e95390. doi:10.1371/journal.pone.0095390

Editor: Danilo Roccatano, Jacobs University Bremen, Germany

Received February 6, 2014; **Accepted** March 26, 2014; **Published** April 18, 2014

Funding: This work was supported by CIC nanoGUNE (www.nanogune.eu) and PRACE (www.prace-ri.eu). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The author has declared that no competing interests exist.

* E-mail: f.corsetti@nanogune.eu

Introduction

The use of first principles atomistic simulations with density-functional theory [1,2] (DFT) has grown from a cottage industry in the early 1990s to a routine and integral part of many contemporary scientific disciplines, at the meeting point between condensed matter physics, physical chemistry, and the new range of nanosciences [3,4]. Potential practitioners have a large number of ready-made codes to choose from (see, e.g., Refs. [5–16]), which distinguish themselves in their licensing models, the range of features they offer, the specifics of the technical implementation, and, generally, where they lie on the (computational) cost–accuracy curve.

An important consideration for all modern DFT codes is their parallel scalability on high-performance computer (HPC) architectures, that open up the possibility of simulating very large physical systems entirely *ab initio*. Consequently, a substantial effort has gone into the development and optimization of many of these codes for the specific purpose of running on massively parallel systems [11,17–28]. Articles describing such developments typically illustrate the scaling performance of the code with an example of strong scaling [9–27], i.e., the wall time speedup obtained for a simulation of fixed size over a range of number of cores. Less frequently, weak scaling performance (i.e., an increase of the problem size proportionally with the number of cores) is also shown [22,25,28].

The use of a strong scaling example can be an effective way of giving a qualitative idea of the parallel efficiency of the code and the scale of problems which can realistically be solved with it. However, there are a number of issues in using the information as it is usually presented for extracting, even approximately, a generalized, quantitative measure of performance, such as could be used to attempt a comparison between codes.

Firstly, the range of cores over which this strong scaling is investigated is not fixed (as must be the case, since time and memory requirements restrict the lower bound, and computational resources the upper bound). The significance of the demonstrated speedup depends crucially on the lower bound of this range; furthermore, the dependence is non-trivial. If we assume a constant rate of loss of efficiency as the parallelization is increased, a speedup of 3.9 when going from 8 to 32 cores should be better than a speedup of 3.8 when going from 2048 to 8192 cores; however, this is obviously not the case, as it is clear from experience that the actual rate increases significantly with the number of cores. Closer comparisons are even harder to judge: is a speedup of 3.8 between 512 and 2048 cores better or worse than a speedup of 3.7 between 1024 and 4096 cores? There is effectively no way to answer this question without making an assumption about how to model parallel performance in general. A well-known and popular, albeit extremely idealized, way to do is by Amdahl's law [29], that describes the overall speedup in terms of the parallelizable fraction of the code P. To the best of our knowledge, only one published strong scaling test for a DFT code [20] has reported on a fitted value for P.

Secondly, there is no standard physical system on which to test strong scaling. From the point of view of the material itself, this is somewhat understandable, as different codes specialize in different areas of modelling; a more fundamental problem, however, is that strong scaling efficiency changes with system size for a given material. Although some studies report system size dependent results [9,12], this is generally not the case. How, then, to compare

between, e.g., a strong scaling test on a 1532-atom carbon nanotube between 2048 and 32768 cores [27], and one on a 1003-atom polyalanine peptide between 512 and 65536 cores [23]?

In this paper, we discuss these issues while reporting on tests of the parallel scaling performance of SIESTA [7], a well-established DFT code based on norm-conserving pseudopotentials [30], a basis of finite-range numerical atomic orbitals (NAOs), and an auxiliary real-space grid for representing the electronic density. The tests are performed on six supercomputers (Table 1), currently forming the network of Tier-0 systems of the Partnership for Advanced Computing in Europe [31] (PRACE). Our aims, therefore, are twofold:

- to give the most up-to-date, comprehensive and reliable results of the timing and scaling of SIESTA on modern HPC systems, so as to allow users of the code to calculate realistic timing estimates over a wide range of number of cores, and therefore plan how to make the best use of their computational resources;
- to propose a simple framework in which to analyze parallel scaling results for all electronic structure codes, arguing in particular for the use of Amdahl's law to quantify strong scaling performance, and for the importance of investigating and reporting this measure as a function of system size.

Computational Methods

Our scaling tests are performed on snapshots of liquid water in cubic boxes with periodic boundary conditions. This is the same system used previously for parallel benchmarking of the Quickstep [9] (CP2K) code; as noted by its authors, liquid water is ideal for this purpose, since boxes of any arbitrary number of molecules can be created while maintaining the same density and cell shape. Furthermore, the lack of crystalline symmetry and the 3D periodicity of the material ensure a sufficiently challenging task that we expect to give a fair idea of worst-case timings for most typical uses of the code, while the presence of a band gap ensures that we do not have to worry about convergence issues arising during the tests.

We simultaneously test weak and strong scaling (again similarly to Ref. [9], by varying both the number of cores, from 32 to 4096 ($N_c = 2^n, 5 \le n \le 12$), and the number of water molecules per core, from 1 to 32 ($N_m/N_c = 2^n, 0 \le n \le 5$). The resulting suite of tests is shown in Fig. 1. The maximum system size tested is of 4096 water molecules (12288 atoms) for all values of N_m/N_c, except for one test of 8192 molecules (24576 atoms) on 8192 cores. We note, however, that due to the limited computational time available on each machine, not all tests are run on all machines. Weak scaling corresponds to moving perpendicular to the N_m/N_c axis, while strong scaling corresponds to moving diagonally. Instead, system size scaling (parallel to the N_m/N_c axis) does not explicitly test parallelization, although it is affected by it, as we shall discuss. The snapshots for all system sizes are extracted from classical molecular dynamics (MD) runs using the TIP4P force field [32] in the GROMACS [33] code, equilibrated to 300 K; the cell shape and volume are kept fixed at the experimental equilibrium density [34] (1.00 g/cm^3).

The tests are performed at the Γ point only (multiple k points being almost embarrassingly parallel), using the semi-local PBE [35] functional for exchange and correlation (xc), a 150 Ry cutoff energy for the real-space auxiliary grid, and, unless otherwise stated, a double-ζ polarized basis [36] (dζ+p), corresponding to 23 NAOs per water molecule; the fraction of occupied eigenstates

Table 1. Specifications for the six PRACE Tier-0 systems.

System	Architecture	Topology	Proc. type	Proc. speed (GHz)	Tot. cores	Tot. nodes	Cores/node	Cores/proc.	Mem./core (GB)
Hermit	Cray XE6	3D torus	AMD Opteron	2.3	113664	3552	32	16	2/4
JUQUEEN	IBM BlueGene/Q	5D torus	IBM PowerPC A2	1.6	458752	28672	16	8	1
FERMI	IBM BlueGene/Q	5D torus	IBM PowerPC A2	1.6	163840	10240	16	8	1
Curie	BullX	Fat tree	Intel SandyBridge	2.7	80640	5040	16	8	4
SuperMUC	IBM iDataPlex	Fat tree	Intel SandyBridge	2.7	147456	9216	16	8	2
MareNostrum	IBM iDataPlex	Fat tree	Intel SandyBridge	2.6	48384	3024	16	8	2

We note that some systems include secondary types of nodes with different specifications; these are not listed here, and are not used for our tests.
doi:10.1371/journal.pone.0095390.t001

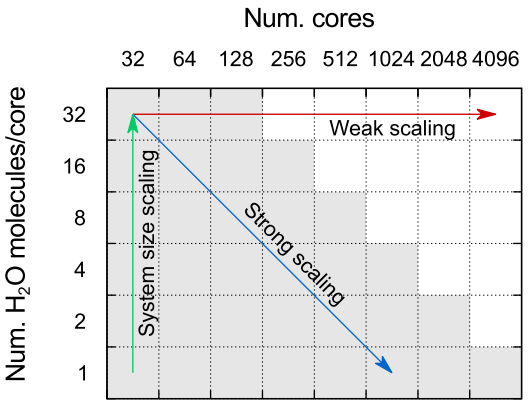

Figure 1. Three types of scaling that can be investigated by systematically varying the number of molecules per core and the number of cores. The shaded cells show the suggested set of tests to perform on a typical HPC system.
doi:10.1371/journal.pone.0095390.g001

is 4/23 ($\sim$17%). All system sizes employ 13 self-consistent field (SCF) iterations to reach convergence.

We use the most recent development version of the code (**siesta-trunk-438**), available on the SIESTA website [37]. The tests are run with the code's default options for diagonalization, employing routines from the ScaLAPACK [38] library: the problem is first transformed from generalized to standard form by Cholesky factorization with the **pdpotrf** and **pdsygst** routines, and then the diagonalization itself is performed with the **pdsyevd** divide-and-conquer routine; finally, the back transform is performed with the **pdtrsm** routine. A 2D block-cyclic data distribution of the matrices is used, with the matrix dimension being an exact multiple of the block size in all cases (tests show the ideal block size to be equal to the number of orbitals per molecule).

We choose the standard solver for our tests, as this is currently by far the most widely used by the SIESTA community; however, we note that several new alternatives are being developed and tested: (i) a solver based on the orbital minimization method (OMM), which has already been demonstrated to exhibit better parallel scaling than explicit diagonalization up to 64 cores [39] (available in the development version of the code), (ii) two new solvers based on ScaLAPACK, the MRRR algorithm [40] and the ELPA library [23] (not yet released), and (iii) a solver based on the pole expansion and selected inversion method [41], specifically designed for massively parallel architectures (not yet released). Finally, the original linear-scaling DFT method implemented in SIESTA is also in the process of being redesigned; in its current implementation it does not scale well on large clusters.

The code was compiled on each of the six machines listed in Table 1 using the native Fortran compiler and optimized linear algebra and communication libraries provided by the system administrators. The Intel compiler and MKL library are used for Intel-based machines (IBM iDataPlex and BullX architectures), the Cray compiler and ACML library for the AMD-based machine (Cray XE6 architecture), and the IBM XL compiler and ESSL library for the IBM PowerPC-based machines (IBM BlueGene/Q architecture). The MPI-2 libraries used are as follows: IBM MPI for SuperMUC, Open MPI for MareNostrum, BullX MPI for Curie, and MPICH2 for Hermit, JUQUEEN and FERMI.

Results and Discussion

Strong scaling

As previously mentioned, Amdahl's law provides a simple model of strong scaling. It states that

$$\mathbb{S}_1(N_c;S)=\frac{t_1}{t_{N_c}}=\frac{1}{S+\frac{1-S}{N_c}}, \tag{1}$$

where $\mathbb{S}_1$ is the speedup obtained on N_c cores with respect to a serial run, t_1 and t_{N_c} are the total execution times in serial and on N_c cores, respectively, and $S=1-P$ is the fraction of the code that is not parallelizable (we prefer using S instead of the more usual P, as the former tends to zero in the limit of ideal scaling). Since it is usually not possible in practice to measure t_1 for large systems, it is useful to define the speedup with respect to a baseline number of cores b instead:

$$\mathbb{S}_b(N_c;S)=\frac{t_b}{t_{N_c}}=\left(\frac{N_c}{b}\right)\frac{S(b-1)+1}{S(N_c-1)+1}. \tag{2}$$

Using this equation, we can fit our strong scaling data over any arbitrary range of number of cores, and obtain a single value S that is in principle independent of this range, and which therefore defines the efficiency of the code for any value of N_c as $1/(1+S(N_c-1))$. The efficiency is invariantly 100% for a serial run, and decreases to zero as $N_c\to\infty$, since the execution time tends to a finite minimum value t_1S.

It is important to note that the conventional interpretation for S and P is necessarily an over-simplification, and should not be taken too literally; nevertheless, Amdahl's law qualitatively reproduces some universal features of strong scaling, and is generally found to provide a good fit to real data. However, such a basic one-parameter model can only describe an average scaling trend, ignoring any system dependent effects that might favour particular values of N_c, e.g., differences in load balancing. Using a homogeneous, scalable system such as liquid water and a regular grid of tests as shown in Fig. 1 can be effective in minimizing these variations, and therefore help to extract clearer general trends.

Using our timing tests for SIESTA on the six different machines, we can analyze the strong scaling of the code for system sizes ranging from 64 to 4096 water molecules; however, we restrict our fitting of S to systems with at least four data points ($\geq$256 molecules). As a representative example, Fig. 2 shows the speedup obtained on SuperMUC (IBM iDataPlex architecture) for four different system sizes, together with the curve fitted from Eq. 2. The resulting S values are robust to fitting over different ranges (within an order of magnitude), as is the trend of decreasing S with increasing system size. It is worth noting that this example clearly illustrates the difficulty in comparing between scaling tests using different ranges of number of cores: despite the steady increase in efficiency revelead by the S values, the speedups shown in the plots appear extremely similar due to the different baselines used.

The top panel of Fig. 3 summarizes the strong scaling results obtained for all six machines: the fitted value of S is given as a function of system size (i.e., the number of water molecules), for systems between 256 and 4096 molecules. Tests for smaller systems (32, 64, and 128 molecules) and larger ones (8192 molecules) are not represented, as there are insufficient data points for a reliable fit.

obtained for the other codes, each available at a single system size. Both the system size and type vary greatly between codes, from an 87-atom 2D system for VASP to a 1532-atom 1D system for QE. Other important factors (k-point sampling, xc functional, basis accuracy, code optimization) are also not controlled for.

Nevertheless, Qbox stands out from all other codes for the impressive strong scaling performance demonstrated, with an S value more than an order of magnitude lower than that obtained by its closest competitor, QE, despite using a smaller system size (1000 atoms). Indeed, Qbox has been developed not only for massively parallel calculations in general, but specifically for running on IBM BlueGene architectures [11,43]; based on these results, it is the only DFT code to have demonstrated the potential to make efficient use ($>50\%$) of the *entirety* of a large BlueGene machine such as JUQUEEN or FERMI for a single Γ-point calculation.

System size scaling and absolute timings

Strong scaling is purely a test of parallel scalability, for which the code is, by definition, taken to be 100% efficient when run in serial. The results presented so far, therefore, contain no information about absolute timings. Although it is convenient to separate these two aspects of the code's performance, we should remember that the execution time is the *only* factor of importance to the end user. Therefore, strong scaling data on its own can sometimes be misleading, as a code that is very fast in serial but which exhibits poor strong scaling might nevertheless achieve a lower execution time on a medium-sized cluster than one that is very slow in serial but with exceptional scalability.

In order to extract a measure of absolute timing from our tests, we need to be able to effectively model system size scaling. For a conventional DFT code that calculates the eigenvalues and eigenvectors of the Kohn-Sham equation [2], either by explicit diagonalization (as we do here) or by an iterative minimization algorithm, it is well known that the calculation time scales cubically with system size (i.e., the number of atoms/molecules/basis orbitals). Linear-scaling methods [44], which make use of approximate spatial truncations based on the principle of electronic nearsightedness [45], are also now well established and have been implemented in a number of popular codes.

The bottom panel of Fig. 4 shows the results of all timing tests performed on two machines, JUQUEEN and SuperMUC; we plot the execution time for all number of cores, extrapolated to that of a single core as $t_{N_c}\mathbb{S}_1$ (from Eq. 1), against the system size (the number of water molecules N_m). The estimated speedup $\mathbb{S}_1$ is obtained for any value of N_m by using the fits of $S(N_m)$ to the data in the top panel of Fig. 3, as described in the previous section. The resulting plot very clearly shows an almost pure cubic scaling with system size for both machines (the linear fits on the log–log scale have a fixed slope of 3). There is an excellent agreement in the extrapolated timings for each system size independently of the number of cores, and, even more encouragingly, our estimate of $\mathbb{S}_1$ appears to be robust even when extrapolating beyond the range of N_m used in the fitting of S.

From these results, we can justify the use of a basic single-parameter model for system size scaling, of the form $t_1=\alpha N_m^3$; lower-order terms are negligible even for the smallest system sizes considered here; this is because all the default routines in SIESTA other than the diagonalization procedure itself are linear-scaling by design. Within an SCF iteration, the contribution from building the sparse Hamiltonian matrix only become comparable to diagonalization for very high values of the cutoff energy defining the real-space grid, or non-local xc functionals such as those including dispersion interactions. We note here that we have also

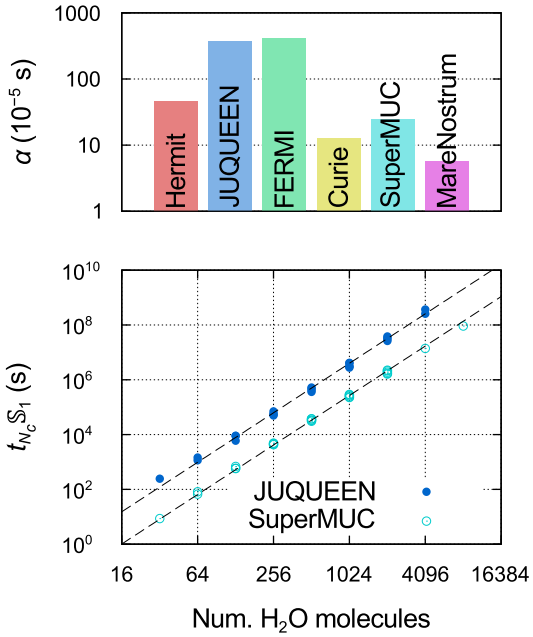

Figure 4. Absolute timings on the six machines. Top panel: prefactor α for the cubic scaling with system size of the execution time in serial for the self-consistent calculation of the liquid water system (13 SCF iterations). Bottom panel: two examples of the fitting of α to absolute timing data, extrapolated for all number of cores to serial timings using Amdahl's law and a fitted analytical expression of the strong scaling performance as a function of system size.
doi:10.1371/journal.pone.0095390.g004

analyzed the strong scaling of individual SIESTA modules, finding diagonalization to be the bottleneck within an SCF iteration, while Hamiltonian construction is very efficient when using the parallelization strategy for the grid operations of Sanz-Navarro *et al.* [21] (accessible via the flag -DBSC_CELLXC at compilation).

The parameter α, obtained by the fits shown in the bottom panel of Fig. 4, can therefore be used to compare the speed of the various machines, independently of differences in scaling performance. The values of α obtained for all six machines are shown in the top panel of Fig. 4. The large variation in α over almost two orders of magnitude is a reflection not simply of the machines' processor speeds (listed in Table 1), but also of numerous other interacting factors, such as the efficiency of the different compilers and libraries. In general, torus machines, which exhibit the best scaling, are predicted to be the slowest in serial, while fat tree machines, which do not scale as well in parallel, are predicted to be the fastest.

We can now calculate a rough estimate of the execution time on each machine for any number of water molecules on any number of cores, by using our fits of the function $S(N_m)$ and the parameter α, and, hence, build up a N_m–N_c 'hase diagram' of the machine with the lowest execution time. This is shown in Fig. 5: the main panel compares real timings, while the inset uses the estimates based on our fits. The agreement is best for large system sizes and number of cores, with some discrepancies appearing for $N_m\leq 256$; this is not surprising, due both to the extrapolation of $S(N_m)$ to low values, and the fact that the timings are very close for more than one machine.

The machines which gives the lowest absolute timings over the entire tested range of N_c are overwhelmingly those with fat tree topologies, despite their inferior strong scaling performance with respect to torus machines. Two large regions can be clearly

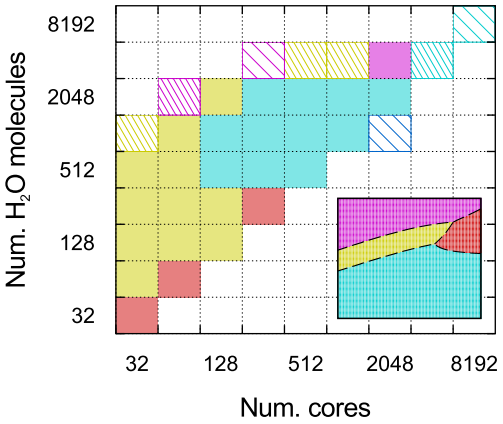

Figure 5. Phase diagram of supercomputers. The machine with the lowest execution time is shown for a given system size and number of cores. The colours used are the same as those shown in the top panel of Fig. 4. Boxes with dashed lines indicate that the data for one or more machines is not available; sparse dashed lines indicate that only one machine was run with these parameters. The inset shows the idealized diagram over the same range, using the timing estimates given by the fits of $S(N_m)$ and α.
doi:10.1371/journal.pone.0095390.g005

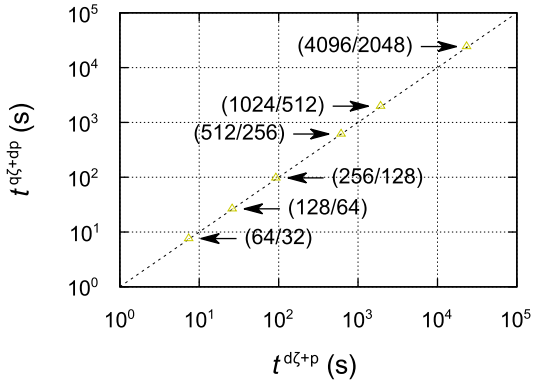

Figure 6. Timing comparison on Curie for two different SIESTA basis sets. Each data point plots the execution time of a particular system size simulated with the $d\zeta+p$ basis (23 NAOs/H_2O molecule) against that of a *different* system size simulated with the $q\zeta+dp$ basis (46 NAOs/H_2O molecule), chosen so that the two systems have the same total number of basis orbitals. The two system sizes are shown in brackets ($d\zeta+p$/$q\zeta+dp$); in each case, both simulations are performed on the same number of cores, equal to the number of molecules in the $q\zeta+dp$ system.
doi:10.1371/journal.pone.0095390.g006

identified: Curie (BullX architecture) is the fastest machine for simulations with $N_c \lesssim 128$, while SuperMUC (IBM iDataPlex architecture) is the fastest above this value. There is some indication, confirmed by the model, that for large system sizes ($N_m \gtrsim 4096$) MareNostrum (IBM iDataPlex architecture) becomes faster than both of these machines (this might seem surprising, since it has the lowest value of α, and, hence, should be the fastest in serial at all system sizes; however, it also exhibits the worst parallel scaling, making it less efficient than other machines for parallel calculations on even very few cores at modest system sizes). It is only for extremely large N_c that the qualitatively different decrease in $S(N_m)$ of the torus machines is predicted to lead to the lowest absolute timings, in particular for Hermit (Cray XE6 architecture), as it has a significantly lower α value than the IBM BlueGene/Q machines.

Our fitted models for the six supercomputers can also be used in a broader context, to estimate the execution time for any typical SIESTA calculation on HPC systems similar to the ones tested here. In fact, since the timing is dominated by the diagonalization procedure, especially for large system sizes, we can base our estimation on only two parameters, the total number of basis orbitals and the number of SCF iterations; we can safely neglect, to a first approximation, other parameters such as the number of electrons and the number and type of ions. This is illustrated in Fig. 6, in which we compare calculations using the standard $d\zeta+p$ basis to ones using a larger $q\zeta+dp$ basis [46] (twice the number of NAOs per water molecule) over a wide range of number of cores; as can be seen, timings on a given number of cores depend only on the total number of basis orbitals, and so a calculation using the larger basis takes the same time as one using the smaller basis with twice the system size. We note that this simple behaviour is due to the use of a solver which computes all eigenvalues by explicit diagonalization. Instead, solvers based on iterative minimization techniques (typically employed by plane-wave codes) scale only quadratically with the number of basis functions [39]; for such codes, we would expect the dependence of $S(N_m)$ on basis size to be non-trivial. Unfortunately, we are not aware of published data for any other DFT code that could help in investigating this issue.

In order to allow SIESTA users to obtain absolute timing estimates for their parallel calculations, we have released a web applet [47] based on the model we have described and the quantitative data obtained from our scaling tests. We also include a version of the applet for offline use in the Supporting Information (Code S1); details of the fits and the final set of parameters for the six machines can be easily found within the code.

Weak scaling

Finally, we briefly discuss the weak scaling behaviour demonstrated by the code. Weak scaling is of most interest to linear-scaling DFT codes [22,25], for which the objective is to obtain a constant time-to-solution as the problem size is increased together with the number of cores (this is also known as Gustafson's law [48]). Cubic-scaling codes, instead, can achieve at best a quadratic weak scaling behaviour, which is rarely investigated [28,39]; nevertheless, it can provide useful information on the limits of efficiency of the code.

We find it convenient to plot the execution time divided by the square of the number of cores, so that ideal weak scaling behaviour will appear flat, analogously to the case of a linear-scaling code. A representative example for one machine, JUQUEEN, is shown in Fig. 7. Surprisingly, we observe better than ideal weak scaling, tending towards ideal as the number of cores is increased. The effect becomes more pronounced as the number of water molecules per core is decreased. These trends are almost perfectly reproduced by the timing estimates provided by our combined modelling of strong scaling and system size scaling.

We can understand this behaviour as a change in efficiency (as defined in the bottom panel of Fig. 3) due to the interplay between the decrease of S with N_m and the increase of N_c. Similarly, it is interesting to note that system size scaling at a fixed number of cores >1 deviates from its ideal cubic behaviour in serial.

If we assume $S(N_m)$ to be of the form $AN_m^{-}B$, it is easily verified from the model that the weak scaling behaviour will tend towards ideal for $B \geq 1$; in the case of JUQUEEN, the fit gives a value of 1.8. This result applies equally to linear- and cubic-scaling codes, when using the appropriate definition of ideal weak scaling

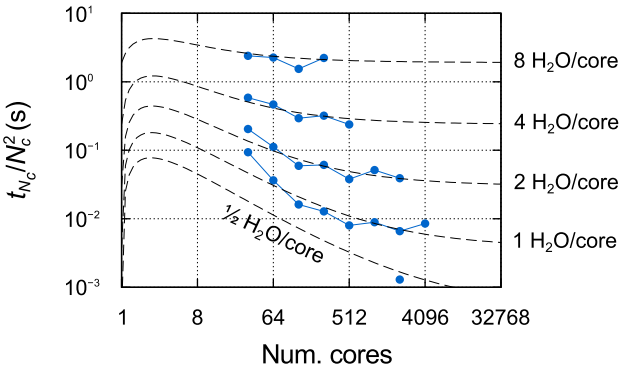

Figure 7. Weak scaling on JUQUEEN for different numbers of water molecules per core. The execution time is divided by the square of the number of cores. The dashed lines show the estimates given by the fits of $S(N_m)$ and α.
doi:10.1371/journal.pone.0095390.g007

for each; indeed, a strikingly similar behaviour is reported for the linear-scaling Conquest code [22]. As noted previously, the three machines with fat tree topologies appear to exhibit a slowing down in the decrease of S with system size; although this should eventually make the weak scaling less than ideal, in practice it is not noticeable within the range of cores considered.

Conclusions

In this paper, we have investigated the performance of the SIESTA code on the six supercomputers of the PRACE Tier-0 network, currently amongst the largest in Europe. We propose a systematic investigation of parallel scaling using self-consistent calculations of snapshots of liquid water, varying both the number of cores on which the simulation is run and the number of water molecules per core; the largest simulation performed in our tests is of 8192 molecules on 8192 cores.

The results are analyzed using Amdahl's law to fit the data for each system size, providing a quantitative estimate of the code's efficiency over all number of cores based on a single parameter S; the scaling performance of the code, therefore, is completely described by the curve of S as a function of system size. We find a qualitative difference in this curve depending on the topology of the connections between nodes in the supercomputer, with machines implementing torus topologies demonstrating a better scalability to large system sizes than those implementing fat tree topologies. Despite this, however, the latter group is shown to give lower absolute timings for almost all simulations within the tested range, as the performance on individual cores is significantly faster; furthermore, such architectures tend to offer a larger amount of memory per core, which can become an important issue either when running on few cores, or as the size of the simulation is increased (the memory requirements scale approximately quadratically with system size).

Combining Amdahl's law for strong scaling with a basic one-parameter model for system size scaling, both of which are fitted to the data provided by our tests, we can calculate a simple estimate of the execution time on a given number of cores for a generic total energy calculation with SIESTA; a new web applet [47] developed in conjunction with the paper allows users of the code to employ this model for planning their projects on parallel architectures. An estimate of the memory requirement per core is also included.

Throughout the paper we have emphasized potential points of comparison with other DFT and electronic structure codes. Investigating and reporting $S(N_m)$ curves for different HPC systems could provide valuable information to practitioners in the field, as well as for the ongoing development of the codes themselves. Care must be taken, however, when interpreting the results of comparisons based on strong scaling data, due to the fundamental differences between codes. Basis sets offer perhaps the most important example: is it meaningful to compare the strong scaling performance of a localized-orbital code and a plane-wave code for the same physical system? It is clear that S varies with basis size, and so is crucially dependent in both cases on the precision level of the calculation; even disregarding the technical challenges involved [46], attempting to equate the two bases is not necessarily appropriate, as the codes are designed from the outset to be used with different aims. For this reason, we suggest that the best approach should not be overly competitive; rather, the objective should be to report on calculations using the typical setup appropriate for each code (e.g., the default $\mathrm{d}\zeta+\mathrm{p}$ basis for SIESTA), or possibly a range of different setups, as this will provide the most useful information for its users.

Supporting Information

Code S1 Bash script for calculating SIESTA timing estimates. These are based on the fits to the data presented in the paper. Instructions for using the script are included as a comment at the start of the code. The script is also available as a web applet [47].
(TXT)

Acknowledgments

We thank Emilio Artacho, Alberto Garcia, and Georg Huhs for useful discussions. We acknowledge PRACE for awarding us access to the following resources: Hermit based in Germany at the High Performance Computing Center Stuttgart (HLRS), JUQUEEN based in Germany at the Jülich Supercomputing Centre, FERMI based in Italy at the CINECA SuperComputing Application and Innovation Department, Curie based in France at the Très Grand Centre de Calcul du CEA (TGCC), SuperMUC based in Germany at the Leibniz Supercomputing Centre, and MareNostrum based in Spain at the Barcelona Supercomputing Center.

Author Contributions

Conceived and designed the experiments: FC. Performed the experiments: FC. Analyzed the data: FC. Wrote the paper: FC.

References

1. Hohenberg P, Kohn W (1964) Inhomogeneous electron gas. Phys Rev 136: B864–B871.
2. Kohn W, Sham LJ (1965) Self-consistent equations including exchange and correlation effects. Phys Rev 140: A1133–A1138.
3. Hafner J, Wolverton C, Ceder G (2006) Toward computational materials design: The impact of density functional theory on materials research. MRS Bull 31: 659–668.
4. Marzari N (2006) Realistic modeling of nanostructures using density functional theory. MRS Bull 31: 681–687.
5. Kresse G, Furthmüller J (1996) Efficient iterative schemes for *ab initio* total-energy calculations using a plane-wave basis set. Phys Rev B 54: 11169–11186.
6. Delley B (2000) From molecules to solids with the DMol^3 approach. J Chem Phys 113: 7756–7764.
7. Soler JM, Artacho E, Gale JD, García A, Junquera J, et al. (2002) The SIESTA method for ab initio order-N materials simulation. J Phys: Condens Matter 14: 2745–2779.
8. Clark SJ, Segall MD, Pickard CJ, Hasnip PJ, Probert MIJ, et al. (2005) First principles methods using CASTEP. Z Kristallogr 220: 567–570.

9. VandeVondele J, Krack M, Mohamed F, Parrinello M, Chassaing T, et al. (2005) Quickstep: Fast and accurate density functional calculations using a mixed Gaussian and plane waves approach. Comput Phys Commun 167: 103–128.
10. Skylaris CK, Haynes PD, Mostofi AA, Payne MC (2005) Introducing ONETEP: Linear-scaling density functional simulations on parallel computers. J Chem Phys 122: 084119.
11. Gygi F (2008) Architecture of Qbox: A scalable first-principles molecular dynamics code. IBM J Res Dev 52: 1–8.
12. Genovese L, Neelov A, Goedecker S, Deutsch T, Ghasemi SA, et al. (2008) Daubechies wavelets as a basis set for density functional pseudopotential calculations. J Chem Phys 129: 014109.
13. Giannozzi P, Baroni S, Bonini N, Calandra M, Car R, et al. (2009) QUANTUM ESPRESSO: A modular and open-source software project for quantum simulations of materials. J Phys: Condens Matter 21: 395502.
14. Blum V, Gehrke R, Hanke F, Havu P, Havu V, et al. (2009) Ab initio molecular simulations with numeric atom-centered orbitals. Comput Phys Commun 180: 2175–2196.
15. Gonze X, Amadon B, Anglade PM, Beuken JM, Bottin F, et al. (2009) ABINIT: First-principles approach of materials and nanosystem properties. Comput Phys Commun 180: 2582–2615.
16. Enkovaara J, Rostgaard C, Mortensen JJ, Chen J, Du lak M, et al. (2010) Electronic structure calculations with GPAW: A real-space implementation of the projector augmented-wave method. J Phys: Condens Matter 22: 253202.
17. Plummer M, Hein J, Guest MF, D'Mellow KJ, Bush IJ, et al. (2006) Terascale materials modelling on high performance system HPCx. J Mater Chem 16: 1885–1893.
18. Bottin F, Leroux S, Knyazev A, Zérah G (2008) Large-scale ab initio calculations based on three levels of parallelization. Comp Mater Sci 42: 329–336.
19. Genovese L, Ospici M, Deutsch T, Méhaut JF, Neelov A, et al. (2009) Density functional theory calculation on many-cores hybrid central processing unit-graphic processing unit architectures. J Chem Phys 131: 034103.
20. Hine NDM, Haynes PD, Mostofi AA, Skylaris CK, Payne MC (2009) Linear-scaling densityfunctional theory with tens of thousands of atoms: Expanding the scope and scale of calculations with ONETEP. Comput Phys Commun 180: 1041–1053.
21. Sanz-Navarro CF, Grima R, García A, Bea EA, Soba A, et al. (2010) An efficient implementation of a QM–MM method in SIESTA. Theor Chem Acc 128: 825–833.
22. Bowler DR, Miyazaki T (2010) Calculations for millions of atoms with density functional theory: Linear scaling shows its potential. J Phys: Condens Matter 22: 074207.
23. Auckenthaler T, Blum V, Bungartz HJ, Huckle T, Johanni R, et al. (2011) Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations. Parallel Comput 37: 783–794.
24. Maniopoulou A, Davidson ER, Grau-Crespo R, Walsh A, Bush IJ, et al. (2012) Introducing k-point parallelism into VASP. Comput Phys Commun 183: 1696–1701.
25. VandeVondele J, Borštnik U, Hutter J (2012) Linear scaling self-consistent field calculations with millions of atoms in the condensed phase. J Chem Theory Comput 8: 3565–3573.
26. Hacene M, Anciaux-Sedrakian A, Rozanska X, Klahr D, Guignon T, et al. (2012) Accelerating VASP electronic structure calculations using graphic processing units. J Comput Chem 33: 2581–2589.
27. Varini N, Ceresoli D, Martin-Samos L, Girotto I, Cavazzoni C (2013) Enhancement of DFTcalculations at petascale: Nuclear magnetic resonance, hybrid density functional theory and Car–Parrinello calculations. Comput Phys Commun 184: 1827–1833.
28. Hakala S, Havu V, Enkovaara J, Nieminen R (2013) Parallel electronic structure calculations using multiple graphics processing units (GPUs). In: Manninen P, Öster P, editors, Applied parallel and scientific computing, Lecture notes in computer science, Heidelberg: Springer, volume 7782. pp. 63–76.
29. Amdahl GM (1967) Validity of the single processor approach to achieving large-scale computing capabilities. In: Proceedings of the April 18–20, 1967, spring joint computer conference. New YorkACM, volume 30 of AFIPS '67 (Spring)483–485.
30. Troullier N, Martins JL (1991) Efficient pseudopotentials for plane-wave calculations. Phys Rev B 43: 1993–2006.
31. PRACE website. Available: http://www.prace-ri.eu/. Accessed 2014 Mar 28.
32. Jorgensen WL, Chandrasekhar J, Madura JD, Impey RW, Klein ML (1983) Comparison of simple potential functions for simulating liquid water. J Chem Phys 79: 926–935.
33. Berendsen HJC, van der Spoel D, van Drunen R (1995) GROMACS: A message-passing parallel molecular dynamics implementation. Comput Phys Commun 91: 43–56.
34. Wagner W, Prus A (2002) The IAPWS formulation 1995 for the thermodynamic properties of ordinary water substance for general and scientific use. J Phys Chem Ref Data 31: 387–535.
35. Perdew JP, Burke K, Ernzerhof M (1996) Generalized gradient approximation made simple. Phys Rev Lett 77: 3865–3868.
36. Wang J, Román-Pérez G, Soler JM, Artacho E, Fernández-Serra MV (2011) Density, structure, and dynamics of water: The effect of van der Waals interactions. J Chem Phys 134: 024516.
37. SIESTA website. Available: http://departments.icmab.es/leem/siesta/. Accessed 2014 Mar 28.
38. Blackford LS, Choi J, Cleary A, D'Azevedo E, Demmel J, et al. (1997) ScaLAPACK users' guide. Philadelphia: Society for Industrial and Applied Mathematics.
39. Corsetti F (2014) The orbital minimization method for electronic structure calculations with finiterange atomic basis sets. Comput Phys Commun 185: 873–883.
40. Antonelli D, Voemel C (2005) LAPACK working note 168: pdsyevr. ScaLAPACK's parallel MRRR algorithm for the symmetric eigenvalue. Technical Report UCB/CSD-05-1399, EECS Department, University of California, Berkeley.
41. Lin L, Chen M, Yang C, He L (2013) Accelerating atomic orbital-based electronic structure calculation via pole expansion and selected inversion. J Phys: Condens Matter 25: 295501.
42. FERMI Software Benchmarks webpage. Available: http://www.hpc.cineca.it/content/fermi-software-benchmarks/. Accessed 2014 Mar 28.
43. Gygi F, Draeger EW, Schulz M, de Supinski BR, Gunnels JA, et al. (2006) Large-scale electronic structure calculations of high-Z metals on the BlueGene/L platform. In: Proceedings of the 2006 ACM/IEEE conference on supercomputing. New YorkACM, SC '0645–52.
44. Bowler DR, Miyazaki T (2012) O(N) methods in electronic structure calculations. Rep Prog Phys 75: 036503.
45. Kohn W (1996) Density functional and density matrix method scaling linearly with the number of atoms. Phys Rev Lett 76: 3168–3171.
46. Corsetti F, Fernández-Serra MV, Soler JM, Artacho E (2013) Optimal finite-range atomic basis sets for liquid water and ice. J Phys: Condens Matter 25: 435504.
47. Siestimator webpage. Available: http://departments.icmab.es/leem/siesta/siestimator/. Accessed 2014 Mar 28.
48. Gustafson JL (1988) Reevaluating Amdahl's law. Commun ACM 31: 532–533.