JCTC Journal of Chemical Theory and Computation

Article

pubs.acs.org/JCTC

Efficient Parallel All-Electron Four-Component Dirac–Kohn–Sham Program Using a Distributed Matrix Approach II

Loriano Storchi,*[,†] Sergio Rampino,[‡] Leonardo Belpassi,[‡] Francesco Tarantelli,[§] and Harry M. Quiney[∥]

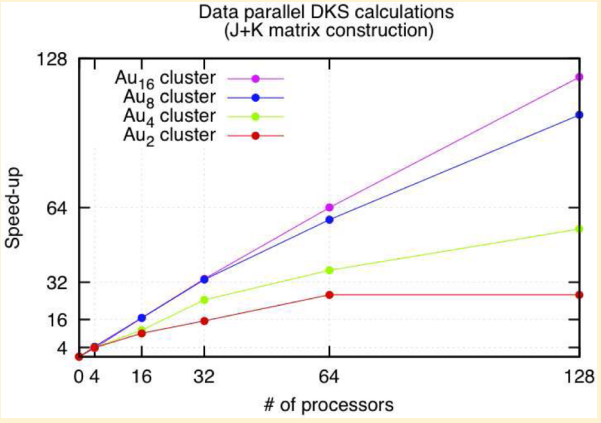
[†]Dipartimento di Farmacia, Università G. d'Annunzio, Via dei Vestini 31, 66100 Chieti, Italy

[‡]Istituto di Tecnologie e Scienze Molecolari del CNR c/o [§]Dipartimento di Chimica, Biologia e Biotecnologie, Università di Perugia, 06123 Perugia, Italy

[∥]ARC Centre of Excellence for Coherent X-ray Science School of Physics, The University of Melbourne, Victoria, 3010, Australia

Supporting Information

ABSTRACT: We propose a new complete memory-distributed algorithm, which significantly improves the parallel implementation of the all-electron four-component Dirac–Kohn–Sham (DKS) module of BERTHA (*J. Chem. Theory Comput.* **2010**, *6*, 384). We devised an original procedure for mapping the DKS matrix between an efficient integral-driven distribution, guided by the structure of specific G-spinor basis sets and by density fitting algorithms, and the two-dimensional block-cyclic distribution scheme required by the ScaLAPACK library employed for the linear algebra operations. This implementation, because of the efficiency in the memory distribution, represents a leap forward in the applicability of the DKS procedure to arbitrarily large molecular systems and its porting on last-generation massively parallel systems. The performance of the code is illustrated by some test calculations on several gold clusters of increasing size. The DKS self-consistent procedure has been explicitly converged for two representative clusters, namely Au_{20} and Au_{34}, for which the density of electronic states is reported and discussed. The largest gold cluster uses more than 39k basis functions and DKS matrices of the order of 23 GB.



I. INTRODUCTION

It is universally recognized that relativistic effects play a crucial role in chemistry, especially for heavy elements.[1−5] The special properties of gold, notably including its catalytic activity[6] both in homogeneous and heterogeneous media, the liquidity of mercury at room temperature, the lead-acid battery deriving most of its voltage from relativistic effects[7] are only a few examples of chemical properties governed by relativity.[8] Among other effects, spin–orbit coupling plays a crucial role in spectroscopy: it not only modifies the energetics of the electronic states but also affects the nature of electronic transitions and enables spin-forbidden excitations. These phenomena are of high relevance in contexts as important in modern technology as, for example, that of dye-sensitized solar cells and organic light emitting diodes, where light absorption and emission quantum yields may be enhanced using complexes containing heavy elements with strong spin forbidden transitions.[9]

For the above reasons, the development of accurate theoretical and computational methods, based on first principles, for the study and characterization of the electronic structure in molecular systems containing heavy atoms is now one of the most important challenges of theoretical chemistry and computational science. The challenge clearly arises from the fact that heavy elements have a very large number of electrons, and both relativistic effects and electron correlation play a crucial role.

A particularly suitable and promising theoretical framework to appropriately treat these systems is the Dirac–Kohn–Sham model (DKS). This combines the rigorous four-component relativistic formalism derived from the Dirac equation with density funcional theory (DFT), which is the method of choice when many electrons are involved, as is the case of large systems containing heavy elements. The relativistic variant of the Hohenberg–Kohn theorem guarantees the formal existence of a current–density functional description of the relativistic systems but does not give any particular hint about how to construct such a functional. So far, the actual implementations of the DKS approach usually resort to the use of standard nonrelativistic density functionals. In general, how to use these functionals may not be trivial and present numerical problems.[10,11] Increasing the efficiency of the DKS method is a crucial step to motivate further research for the development of new functionals which properly use the relativistic four-current density as the basic variable.

Important and significant computational progress has been made in recent years.[12−16] Our specific implementation of the

Received: August 25, 2013
Published: November 19, 2013

four-component DKS theory in the package BERTHA[17] is based on the electron-density fitting approach that is already widely used in the nonrelativistic context (see also refs 18−20). We have implemented the variational Coulomb fitting approach in our DKS method,[21] with further enhancements resulting from the use of the Poisson equation in the evaluation of the integrals,[22−24] and also from the extension of the density fitting approach to the computation of the exchange-correlation term.[25] As shown very recently by Kelley and Shiozaki,[26] these density fitting techniques appear to be suitable for the implementation of Dirac−Hartree−Fock including Coulomb, Gaunt, and full Breit interaction.

The above-mentioned algorithmic advances have represented a leap forward of several orders of magnitude in the performance of the four-component DKS approach but left two big hurdles untouched on the way toward truly large-scale applications: one is the fact that the computational bottleneck is progressively shifted toward the linear algebra operations and, especially, matrix diagonalization; the other, related, problem is the "memory bottleneck". The huge basis sets required for the accurate all-electron treatment of many heavy atoms at once lead to matrix sizes that simply cannot be handled on a typical computer. It seems clear that only a suitably designed data-parallel, fully distributed-memory approach can be of help on both counts. This kind of data-parallel paradigm is in fact in line with the current architectural trend toward many cores and massively parallel systems, where the computational resources are made of many compute units (cores) having each a relatively small amount of memory (we think here for instance of Nvidia's CUDA,[27] Intel Many Integrated Core Architecture,[28] or the IBM BlueGene project[29]). It thus appears important to achieve the right mix of scalability and extensive memory distribution.

In our first parallel implementation,[30] we adopted a sort of master−slave paradigm which only partially solved the memory bottleneck problem. The effective memory distribution was achieved thanks to a specific feature of the SGI Altix 4700 system where we developed and tested the implementation. This system offered a fast and transparent global memory mechanism, the SGI NUMAflex.[31] Here we present and discuss a new implementation that poses the basis for the development of a complete and explicit memory distribution scheme with the aim of fully overcoming all the limitations mentioned above. The parallel software framework and library we used are the widely available (and often optimized) Message Passing Interface (MPI)[32] and ScaLAPACK libraries.[33]

In section II we briefly summarize the SCF procedure in BERTHA, as well as the master−slave paradigm previously implemented. We then describe in detail the new data-distribution scheme. In section III, we discuss the efficiency of our new approach as resulting from some large all-electron calculations on several gold clusters of increasing size. Finally, as an interesting benchmark application, we reach the convergence of the SCF procedure for two large gold clusters, Au_{20} and Au_{34}, using large basis set and analyze their electronic structure through an analysis of the respective density of states.

II. PARALLELIZATION STRATEGY

Recently, we reviewed the theoretical and computational advances made in the last years in our DKS implementation, especially underlining the main peculiarities of the formalism in relation to the density fitting procedure.[17] Further details of our implementation can be found in the original papers.[21,24,25,34−38] For the reader's convenience, a brief summary follows, focusing in particular on the steps making up a typical SCF iteration and their computational costs.

In BERTHA, the spinor solutions of the DKS equation are expanded as a linear combination of Gaussian G-spinor basis functions. This allows for an exact evaluation of the density elements as a finite linear combination of standard Hermite Gaussian-type functions (HGTF). This formulation enables a highly efficient analytic evaluation of all the required multicenter G-spinor interaction integrals. Notably, the implementation takes advantage of the relativistic generalization of the J-matrix algorithm, which allowed for a particularly efficient and straightforward implementation of density fitting procedures based on the Coulomb metric. This efficiency is further enhanced by the choice of the fitting basis set consisting of sets of primitive HGTFs of common exponents and spanning all angular momenta from zero to the target values (for all details see ref 21). This aspect was found to be particularly effective when high angular momenta are necessary, which is indeed the case of the molecular systems containing heavy elements. The fitted density is directly used for the calculation of the exchange-correlation potential in the limit of the generalized gradient approximation. Using such a combined approach, the Coulomb and exchange-correlation contribution to the DKS Hamiltonian, which used to largely dominate the whole computational cost, can be formed in a single step. The other contributions to the DKS matrix, namely the kinetic energy and electron−nuclei interaction terms, are computed once at the beginning and do not change during the SCF procedure. The reduction of the computational cost afforded by the density fitting scheme was shown to be dramatic. Besides reducing the scaling power for the construction of the DKS matrix from $O(N^4)$ to $O(N^3)$, we gained an enormous reduction in the prefactor, without appreciable effects on the accuracy.[25] We thus achieved the notable result that the DKS matrix construction presents a computational cost that is of the same order of magnitude as the linear algebra operations typically required in the SCF procedure (i.e., matrix−matrix multiplications and matrix diagonalization). Both of these contributions present a formal scaling of $O(N^3)$.

Typically, in a SCF iteration, the Coulomb and exchange-correlation matrix contribution ("J+K matrix" from now on) takes about a third of the whole iteration time, while the rest of the time is essentially related to a few linear algebra operations ("Linear Algebra" from now on). These include clearly the DKS matrix diagonalization, requiring half of the time, a level-shifting phase which involves a double matrix multiplication transforming the DKS matrix from basis function space to spinor space, and finally a matrix multiplication necessary to obtain the density matrix starting from the occupied positive-energy spinors. The parallelization effort targets these time-consuming phases, leaving unparallelized the smallest part possible. Thus, we parallelized the above cited steps, leaving unparallelized only a small part (less than 2% of the total SCF iteration time) that is related to the density fitting procedure, together with the HGTF expansion of the density. We will label, from now on, this part of the code as "serial phase".

1. Master−Slave Parallelization Strategy. In a previous paper,[30] we presented in detail a master−slave (MS) parallelization scheme. We shall here briefly recall its main aspects in order to provide the context in which the new data-parallel (DP) scheme has been developed.

In the MS scheme, we adopted a paradigm which only partially solved the memory bottleneck problem. In that model, one of the concurrent processes, the master, allocates in fast memory the entire arrays required for the calculation, specifically: the G-spinor basis overlap, density, and **J+K** matrices, while all other processes share the computation burden. For large memory requirements, this model can only work on parallel systems offering a node with a large amount of memory or a fast transparent global memory mechanism. The latter is the case of the SGI NUMAflex architecture[31] mentioned earlier. In the MS approach, the master process carries out the "serial phase" of the SCF iteration, while all the concurrent processes share the burden of the other calculation phases and have to allocate only some temporary small arrays when needed. This approach can be summarized by the following pseudocode:

```
IF ( my_rank .eq. master ) THEN
  RECV FROM ANY_SLAVES (blockdim, block_position)
  IF (blockdim < 0) then
    EXIT
  ELSE
    RECV FROM slave (block)
    copy block into J+K matrix
  ENDIF
ELSE
  DO block_num = 1, max_nom_of_block

    IF ((my_rank+((num_of_processors)*my_block_num)) == block_num)
       my_block_num = my_block_num + 1
       ALLOCATE block
       COMPUTE block
       SEND (blockdim, block_position)
       SEND (block)
    END IF

  ENDDO

  SEND (blockdim = -1)
  EXIT
END IF
Distribute J+K matrix

Linear algebra using ScaLAPACK
(Level Shift, Diagonalization, Density)
```

The parallelization of the **J+K** matrix construction is based on the assignment of matrix blocks to be computed to each available process except the master. The matrix block structure is naturally dictated by the grouping of G-spinor basis functions in sets characterized by common origin and angular momentum (see also ref 37). The master process needs to broadcast only a small vector of double precision numbers, of size equal to the number of fitting basis fuctions, representing the sum of the Coulomb and exchange-correlation potential in the Coulomb metric (for details see ref 25). Each slave process allocates only a local small array where it stores the **J+K** matrix block it computes and, after the computation is done, sends the block, together with its dimensions and position within the global **J+K** matrix, to the master. The master stores each block, as soon as it arrives, in the proper location of the allocated **J+K** matrix. This MS approach guarantees a good overlap between communication and computation during the "J+K matrix" phase. In addition, the fact that each slave computes a small portion of the **J+K** matrix at a time, provides also a good cache reuse.[30] After this step, the **J+K** matrix needs to be distributed among all the processes, using the so-called two-dimensional block-cyclic distribution.[33] This distribution is mandatory to use the various ScaLAPACK routines needed to perform the linear algebra operations in parallel.

The ScaLAPACK routines we used for the DKS program are PZHEMM in the "level shift" phase, PZGEMM in the "density" phase, and finally PZHEGVX to carry out the complex DKS matrix diagonalization. At the end, we collect on the master the density matrix. Both the collection and distribution of the above matrices are performed by routines implemented by us.[30] Thus, apart from the internal communication activity of the ScaLAPACK routines, there are three explicit communication operations: the slave-to-master transfer of the **J+K** matrix blocks, the initial distribution of the DKS matrix, and the final gathering of the resulting density matrix. Those communications take a running time that is, for a given matrix dimension, almost independent of the amount of processors involved.[30] Indeed the performance obtained with this approach appears to converge to more than 80% of the theoretical maximum on 128 processors, and it turns out to be about 60% of the limit value for an infinite number of processors (when the execution time reduces to that of the unparallelized portion).[30]

As mentioned above, however, the MS scheme obviously imposes, by construction, a limit to the size of the system that can be handled and some specific constraints to the hardware platform. In the most common situation, a "fat" node, working as master and capable of allocating the memory required for the whole arrays discussed above, is clearly necessary. We will show in the following that an approach that avoids to allocate this big amount of memory is possible. This approach will be most suited to improve the performance of the DKS module of BERTHA in terms of its memory usage and portability on state-of-the-art massively parallel architectures.

2. New Data Parallel Strategy. As we just mentioned, the main drawback of the MS parallelization scheme is that it assumes that the master process can functionally allocate any amount of memory as required. We have now overcome this limitation by adopting a complete DP approach, where each process performs a similar task on a different set of data, and there is no distinction between master and slaves. In this new scheme, the **J+K** matrix is never stored in a single process, but is, at all times, distributed among the concurrent processes. Because in the earlier MS implementation the master process had only the role of controlling the required arrays and dispatching the workload and did not share the computation burden, a further benefit of the new DP scheme is that no process is lost to the computation. This advantage, of course especially felt for small numbers of processes, is augmented by the better cache reuse resulting from the reduced size of the local arrays. The new procedure for the **J+K** matrix computation can be summarized by the following pseudocode:

Table I. Times in Seconds for the DISTRIBUTESPARSE (A) and DISTRIBUTE (B) Routines as a Function of Matrix Size and Number of Processors[a]

	matrix dimension							
	1560		3120		6240		12480	
no. of processors	A	B	A	B	A	B	A	B
4	0.11	0.12	0.43	0.47	1.71	2.35	2.86	8.81
16	0.06	0.11	0.17	0.48	0.56	1.87	2.57	7.31
32	0.06	0.10	0.15	0.43	0.46	1.72	1.92	6.91
64	0.09	0.10	0.26	0.45	0.64	1.78	2.05	7.08
128	0.15	0.11	0.51	0.46	1.09	1.85	2.76	7.47

[a]Data obtained on IBM-SP6 and using MPI library. See section III for details.

```
DO block_num = 1, max_nom_of_block
  IF ((my_rank+((num_of_processors)*my_block_num)) == block_num)
    my_block_num = my_block_num + 1
    ALLOCATE block
    COMPUTE block
  END IF
ENDDO

CALL DISTRIBUTESPARSE

Linear algebra using SCALAPACK
(Level Shift, Diagonalization, Density)
```

Each process computes a balanced subset of blocks of the **J+K** matrix and stores the result in a local array. This approach does not require the collection of the matrix on a single process and completely eliminates data communication during the computation of the **J+K** matrix. Indeed, apart from the implicit internal communication activity of the ScaLAPACK routines, there only are four explicit communication operations. First we need to initially distribute the overlap matrix; this communication takes place only once at the beginning of the entire DKS procedure. Second, the root process broadcasts the vector of fitting coefficients at the outset of the J+K matrix step. This is a small vector whose dimensions are obviously related to the fitting basis set size selected. The third communication step takes place immediately after the **J+K** matrix computation. This is an all-to-all communication performed by the DISTRIBUTESPARSE routine we are about to describe. The total amount of data transferred in this step is in any case not larger than the **J+K** matrix dimension. Moreover, as shown in Table I, the time required for this communication is largely independent of the number of processors involved. Finally, after the "density" step,[30] we need to perform a final gather of the density matrix.

3. DISTRIBUTESPARSE Routine. At the end of the **J+K** matrix computation, the matrix is completely distributed among all processes, in accord with the matrix block structure dictated by the grouping of G-spinor basis functions in sets characterized by common origin and angular momentum (integral driven distribution),[37] as recalled earlier. However, this distribution is not suitable for use by the subsequent ScaLAPACK routines and so the matrix needs to be redistributed according to the appropriate two-dimensional block-cyclic distribution. To perform this redistribution we implemented a specific routine (DISTRITUTESPARSE) which maps the two distribution patterns: this step constitutes the core of our new implementation and will be described in the following.

The DISTRIBUTESPARSE algorithm we have implemented permits the efficient remapping of an arbitrary distribution scheme of a matrix among processors onto the block-cyclic distribution required by the linear algebra routines of the ScaLAPACK library. According to this two-dimensional block-cyclic data distribution (BCDD), a dense matrix is decomposed into blocks ("ScaLAPACK blocks") of suitable size ("ScaLAPACK block-size"), which are then uniformly distributed along each dimension of the process grid, so that every process owns a subset of them. There is not a simple and predictable relation between the BCDD and the integral driven distribution, indeed the latter is dictated by the grouping of G-spinor basis functions in sets characterized by common origin and angular momentum, thus is a function of the basis set used as well as of the molecular system and of the number of processes. The BCDD is instead a function of the twodimensional virtual process topology used and of the ScaLAPACK block-size. The only possibility, with the present scheme, to directly optimize the mapping of the two distributions given an input basis set and molecular system is to modify the process topology and/or the ScaLAPACK block-size, but this must be done for each specific case.

As first step of DISTRIBUTESPARSE, each process must share, via an MPI broadcast, some information about the blocks of the **J+K** matrix it currently owns as a result of the previous computation. This information includes the set of processes it needs to communicate with (in order to redistribute its blocks according to the BCDD), data dimensions, etc. Once this initialization phase is finished, each process packs the computed **J+K** matrix elements into sets, according to the ScaLAPACK blocks the elements belong to. The result of this step is a set of blocks or sub-blocks each of which must eventually be owned by a specific process of the ScaLAPACK grid. After this packing step, data are ready to be sent to the legitimate owner and the effective communication of the matrix blocks is carried out through a series of MPI point-to-point primitives[32] that, when possible, are performed in parallel. Obviously the amount of data exchanged cannot be easily predicted, being a complex function of the problem size (molecular system, basis set used), as well as of the processor grid shape[33] adopted and obviously of the ScaLAPACK block-size chosen. The DISTRIBUTESPARSE routine is much more complex than the DISTRIBUTE(mat,rank) routine of the previous MS approach. In the latter, the entire matrix mat is stored in the local memory of a single process rank that distributes it among all processes according to the BCDD scheme. The DISTRIBUTESPARSE routine is instead semantically equivalent to a long sequence of DISTRIBUTE(mat,rank) calls, where each process rank in

dx.doi.org/10.1021/ct400752s | J. Chem. Theory Comput. 2013, 9, 5356−5364

turn calls the DISTRIBUTE routine for each one of the J+K matrix blocks it has computed.

Although, as mentioned above, the communication activity related to the DISTRIBUTESPARSE routine requires the broadcast of some mapping information, this is of course marginal compared to the actual matrix block transfer and, as a result, the total communication time does not display any appreciable dependence on the number of processes involved and is very efficient. This is highlighted in Table I, where one can see that the overall matrix redistribution time is less than 3 s, regardless of the number of processors involved, up to the largest case treated, which involves a 12 480 × 12 480 matrix of double precision complex numbers.

It is also worth noticing, as the table also shows, that the DISTRIBUTESPARSE implementation improves over the previous MS scheme (DISTRIBUTE function[30]) in which the BCDD was carried out starting from a matrix entirely allocated on the master process. This should be expected, because the overall amount of data transferred in the new scheme is of course normally smaller than the whole DKS matrix that was distributed in the MS model. This is an additional welcome feature.

Though the overall performance of the DISTRIBUTESPARSE routine improves over the DISTRIBUTE one, it may be noticed that, for the matrices of size 1560 and 3120 using 128 processors, the former routine is slower than the latter (Table I). This effect is explained by the fact that, when the amount of data to be sent is too small, the overall performance of the DISTRIBUTESPARSE routine is mainly driven by the broadcasts needed during the initialization phase.

III. DISCUSSION

To put the new DP implementation of our DKS module to the test, we performed several computations for a set of representative gold clusters: Au_2, Au_4, Au_8, Au_{16}. The corresponding DKS matrix sizes are 1560 for Au_2 (37.1 MB), 3120 for Au_4 (148.5 MB), 6240 for Au_8 (594.1 MB), and 12480 for Au_{16} (2.3 GB). The large component of the G-spinor basis set on each gold atom was derived by decontracting the double-z-quality basis set of Dyall[39] (22s 19p 12d 8f). The corresponding small-component basis was generated using the restricted kinetic balance relation.[40] We used the BLYP exchange-correlation functional, made of the Becke 1988 exchange functional (B88)[41] plus the Lee–Yang–Parr (LYP) correlation functional.[42] As auxiliary basis set, we used the HGTF basis optimized by us and denoted as B20 in ref 21. A numerical integration grid[34] has been employed with 61 200 grid points for each gold atom. Moreover we do not make use of any cutoff scheme of cutoff threshold of analytical two-electron integrals.

We report our results referring directly to the grid of processors used rather than the total number of processors. This is more appropriate to the ScaLAPACK environment, which imposes a mapping of the total number of processes, P, into a two-dimensional rectangular grid. This grid ($P_r \times P_c$) will have P_r rows and P_c columns, with $P_rP_c = P$. The importance of the shape of the grid, as well as of the ScaLAPACK block-size driving the block-cyclic decomposition, has been discussed in detail in our previous work.[30] The grid shape affects appreciably the performance of the linear algebra routines and different routines may be differently influenced depending on the block size, number of processes, and size of molecular systems. In our experience, we found a spread in performance of up to 50%, with rectangular grids appearing to be relatively unfavorable for the diagonalization step (the computationally most expensive step) compared to square grids. Therefore, we prefer a square processor arrangement when possible and choose $P_r < P_c$ otherwise. The block dimension used is 32. Before we proceed, it is necessary to add a note about the system hardware and software we used to test our code. All the results reported have been obtained using an IBM SP Power 6 installed at CINECA (i.e., IBM P6-575 Infiniband Cluster equipped with IBM Power6 4.7 GHz CPU and Infiniband x4 DDR networking), with the IBM implementation of the MPI library and the ScaLAPACK library. An initial porting of our code to a massively parallel architecture (the FERMI BluGene/Q system) is encouraging and preliminary results will also be briefly presented here.

In Table II, we report the speedup of the **J+K** matrix step, both for the MS and the DP approaches. As can be seen, both

Table II. J+K Matrix Construction Speedup, for the Master–Slave (MS) and Data-Parallel (DP) Approaches As a Function of the Processor Grid Used[a]

cluster	parallelization scheme	2 × 2	4 × 4	4 × 8	8 × 8	8 × 16
Au_2	MS	2.9	9.6	14.5	26.6	26.6
	DP	3.9	10.1	15.4	26.6	26.6
Au_4	MS	2.9	10.6	24.0	35.6	54.6
	DP	3.9	11.4	24.4	37.1	54.8
Au_8	MS	3.1	15.4	32.3	58.7	100.5
	DP	4.4	16.7	33.1	58.8	103.8
Au_{16}	MS	2.8	15.2	31.3	62.5	119.5
	DP	3.7	16.7	33.4	64.0	120.0

[a]All the tests have been performed on the IBM-SP6 at CINECA.

schemes display an excellent performance, which also testifies of the good load balance implicit in their design. The Au_2 and Au_4 cases are exceptional in that, because of their small size, they rapidly reach a speedup limit. The DP approach, besides achieving the complete memory distribution it has primarily been designed for, is also slightly superior, as expected, especially when running on few nodes, due to the extra processor gained to the computation. As the number of processes increases, furthermore, the DP scheme still retains a slight edge essentially because of the negligible communication activity. We even observe cases of superlinear performance when the small size of the local arrays permits improved cache reuse to prevail over other factors.

We mention that the good load balancing of the **J+K** matrix step for gold cluster calculations can be achieved also in the case of molecules containing different atoms. We performed some calculations on the organometallic system $[(Ph_3P)Au(C_2H_2)]^+$, and the results are presented as Supporting Information. These tests used two different G-spinor basis sets derived by decontracting a double-ζ and a triple-ζ quality basis set. The results show no appreciable deviance in terms of speedup.

The overall performance of the two parallelization strategies is summarized in Figure 1 where we report the overall speedup of a SCF iteration obtained using both the MS and the new DP scheme, together with the maximum theoretical speedup computed according to Amdahl's law[43] taking into account the unparallelized fraction. We can see that the DP design increases the overall performance of our parallel DKS module, by up to 15% depending on the number of processors. Table III

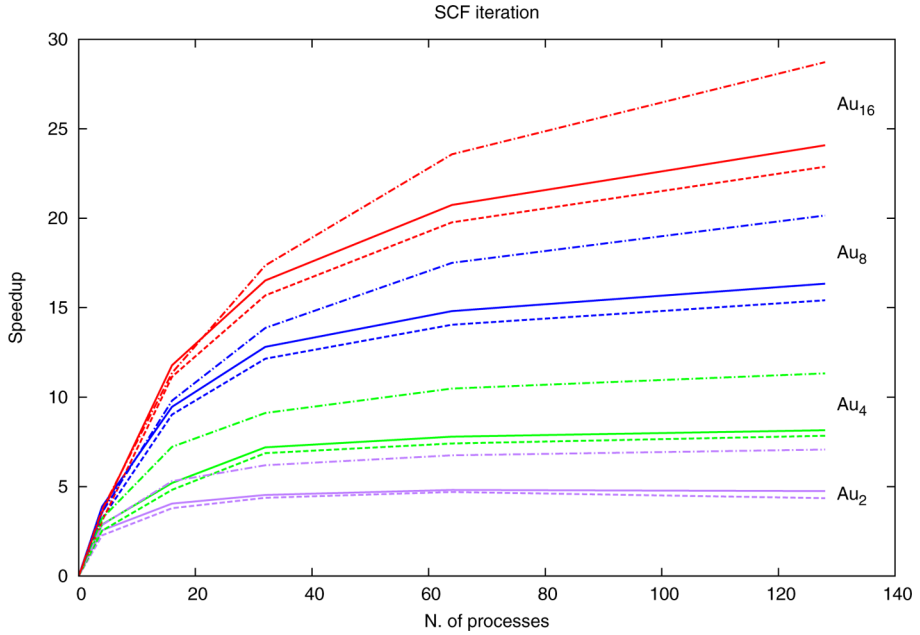


Figure 1. Overall speedup for the MS (dashed lines) and DP (solid lines) approaches for the various gold clusters investigated: purple for Au_2, green for Au_4, blue for Au_8, and red for Au_{16}. The maximum attainable speedup resulting from Amdahl's law is in each case also reported (dotted−dashed lines).

Table III. Elapsed Real Times (s) for the Various Phases of the DKS Calculations on Some Gold Clusters As a Function of the Number of Concurrent Processes Employed (Indicated by the ScaLAPACK Grid Shape)[a]

			MasterSlave					Data Parallel				
Cluster	Step	Serial	2 × 2	4 × 4	4 × 8	8 × 8	8 × 16	2 × 2	4 × 4	4 × 8	8 × 8	8 × 16
Au_2	J+K matrix	14.34	5.00	1.49	0.99	0.54	0.54	3.67	1.42	0.93	0.54	0.54
	Linear Algebra	25.44	4.03	2.03	1.76	1.76	1.79	3.72	1.97	1.72	1.76	1.83
	Serial phase	3.79	3.23	3.84	3.63	3.63	4.07	3.39	3.40	3.41	3.41	3.41
	Total iteration	43.57	12.26	7.36	6.38	5.93	6.40	10.78	6.79	6.06	5.71	5.78
Au_4	J+K matrix	105.30	35.87	9.92	4.39	2.96	1.93	26.71	9.24	4.31	2.84	1.92
	Linear Algebra	202.56	24.76	13.49	7.93	7.25	6.79	25.35	12.74	7.63	7.04	6.83
	Serial phase	15.59	15.51	16.48	15.67	15.73	15.82	14.73	15.00	14.79	14.82	14.85
	Total iteration	323.45	76.14	39.89	27.99	25.94	24.54	66.79	36.98	26.73	24.70	23.60
Au_8	J+K matrix	885.63	282.02	57.44	27.44	15.08	8.81	202.57	53.11	26.74	15.05	8.53
	Linear Algebra	2051.89	198.09	64.13	43.30	35.34	30.21	185.96	64.90	41.92	34.39	29.37
	Serial phase	75.04	75.11	75.67	75.79	76.32	76.53	70.17	70.26	70.30	70.81	71.04
	Total iteration	3012.56	555.22	197.24	146.53	126.74	115.55	458.70	188.27	138.96	120.25	108.94
Au_{16}	J+K matrix	6563.00	2348.56	432.19	209.46	105.03	54.92	1776.84	393.91	196.31	102.51	54.70
	Linear Algebra	13681.08	1915.55	451.51	298.43	215.36	165.83	1899.12	447.61	293.58	210.33	161.88
	Serial phase	390.60	390.98	405.23	406.97	405.72	406.84	377.90	377.79	379.07	379.30	379.73
	Total iteration	20634.68	4655.09	1288.93	914.86	726.11	627.59	4053.86	1219.31	868.96	692.14	596.31

[a]Data are reported for both the MS and DP implementations, as well as for the serial code. The linear algebra times reported are the sum of the three ScaLAPACK driven steps (level shift, diagonalization, and density) and include the communication time for the block−cyclic array distribution.

reports the elapsed times of the different phases of the SCF iterations, for the various gold clusters and different numbers of processors employed (shown in their P_rP_c ScaLAPACK arrangement). We note here that the linear algebra operations perform almost identically in both the MS and DP cases. The small difference is mainly due to the communication needed for the matrix distribution, which we include in this entry and, as discussed earlier, is more efficient in the DISTRIBUTESPARSE function of the DP approach. As a general remark about ScaLAPACK, in-depth analysis using the SCALASCA performance toolset[44] shows that even if they generally present a good load balancing (depending on matrix dimension, ScaLAPACK block-size, and virtual process topology used) communication represents a bottleneck. In fact, most of the communication in the present parallelization scheme is due to ScaLAPACK itself, and essetially to the diagonalization routine. Therefore, the linear algebra steps represent essentially the sole efficiency limiting factor in porting the code to high latency and low bandwidth network clusters.

Interestingly, a further somewhat surprising benefit of the homogeneous array distribution of the DP implementation is that the "serial phase" of the computation appears to be consistently slightly faster (by about 10%) than what is measured for the serial code and for the MS approach. We attribute this to the more efficient overall process management

resulting from the much reduced memory allocated by the ex-master process carrying out the serial fraction.

We mention here that a preliminary porting, with no optimization attempt, of our DP code to the FERMI BlueGene/Q system at CINECA (equipped with IBM PowerA2 1.6 GHz CPU and 11 links → 5D Torus Network interface) also gave encouraging results. Using 128 processes for the Au_{16} cluster the performance of the **J+K** matrix step was 99.7% of that of the 8 × 16 grid on the SP Power 6, while the ScaLAPACK portion performed 24% faster. The porting also displayed good scalability properties, as the **J+K** matrix step on 256 processes (a 16 × 16 grid) turned out to be 75% faster than on 128. (We cannot give absolute speedup ratios as we could not run the serial code on the FERMI BluGene/Q.) This shows that the new implementation is, as designed, effectively "open-ended" in terms of memory demands and can successfully be ported to massively parallel architectures with small memory-per-core figures.

We performed several numerical tests to check the effective numerical stability of the code. These show that, in the SCF procedure, the program can achieve convergence on the energy to within 10^{-9} hartree for the Au_8 and Au_{16} clusters, both of them using a 4 × 4 two-dimensional virtual process grid. The convergence on the density (maximum absolute element of the density matrix difference between the last two cycles) is 10^{-6}. We did not register any numerical instability also for the organometallic compound $[(Ph_3P)Au(C_2H_2)]^+$ with convergence on the energy to 10^{-8} hartree, again on a 4 × 4 grid. This efficiency and numerical stability, unprecedented in the 4-component DKS framework, promises for the first time substantial reward for serious efforts toward new theoretical and computational achievements, including the implementation of automatic geometry optimization techniques.

A. Benchmark Applications: Density of States in Au_{20} and Au_{34}. In order to prove the effective usability and numerical stability of our new parallel implementation, we carried out two complete SCF calculations on two representative gold clusters, namely Au_{20} and Au_{34} with a large basis set. The structures of the clusters were preliminarly optimized using the zero order relativistic approximation (ZORA) with small core and a QZ4P basis set as implemented in the ADF package.[45−48] The large component of the G-spinor basis set used in BERTHA was obtained by decontracting the Dyall basis set for gold of triple-*z* quality (29s, 24p, 15d, 11f, 3g, 1h).[39] This is a larger basis set than that used in the Au_2−Au_{16} test calculations. The corresponding small component basis was generated using the restricted kinetic balance relation.[40] In the largest case, this results in a DKS matrix of dimension 39576 (23.3 GB). We used the BLYP density functional. A numerical integration grid[34] has been employed with 61 200 grid points for each gold atom. The calculations were carried out on the IBM SP6 machine cited above, using 64 processors and with a total energy convergence threshold of 10^{-6} hartree. The time required to complete a single SCF iteration in the case of Au_{34} was about 4.5 h and each single-point DKS calculation required about 25 iterations to reach convergence.

In Figure 2, we report the electronic structure of Au_{20} in terms of the density of states (DOS) resulting from the 220 positive-energy electronic states with highest energy. These correlate with the electrons in the *ds* shell of the isolated atoms. We include for comparison the results obtained by three different calculations using ADF, in combination with the nonrelativistic Hamiltonian (n.r.), the scalar relativistic ZORA Hamiltonian, and spin−orbit ZORA Hamiltonian. In all cases, we treat explicitly all electrons in the self-consistent field procedure in combination with the TZ2P basis set.[45−47] In all cases, a Gaussian broadening of 0.05 eV has been used. The computed DOS shows, as one may expect, a strong dependence on the theory used. The n.r. Hamiltonian yields a pattern of states that is in evident qualitative mismatch with that of the reference DKS calculation: the presence of states at energies around −4.5 eV is the clear evidence of an artificial destabilization of the valence electrons that arises from neglect of relativistic effects. If these effects are included, even in the approximate scalar fashion provided by the scaled ZORA Hamiltonian, the agreement with DKS improves and the satellite states (at −4.5 eV) disappear. The remaining mismatch may be largely attributed to the spin−orbit coupling effect that is indeed absent in the scalar ZORA treatment. This is confirmed by the comparison with the spin−orbit ZORA calculation which is indeed in very good agreement with our DKS results.

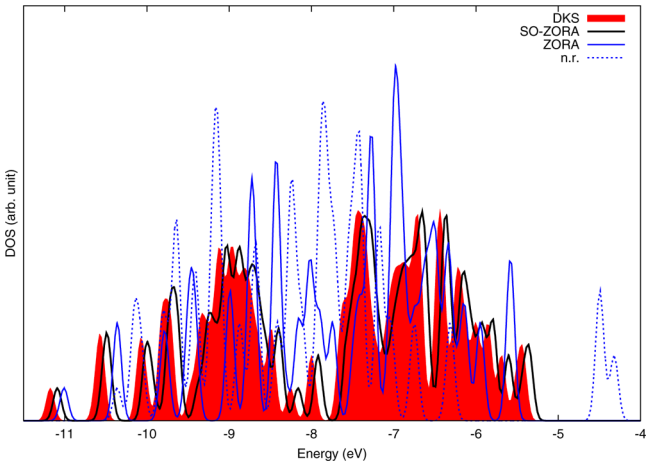


Figure 2. Density of states (DOS) of the Au_{20} cluster using different Hamiltonians and including our DKS calculation. A Gaussian broadening of 0.05 eV is used.

In Figure 3, we give the comparison of the DOS in the Au_{20} and Au_{34} clusters. The cluster structures are reported in the upper panel of the figure. The DOS of Au_{34} presents a more uniform distribution than that of the Au_{20} case. This is strictly related to its amorphous geometry. Such a structural/symmetry dependence of the electronic state distribution is well-known in the literature and has been be used to identify the cluster geometries with the aid of spectroscopy measurements.[49]

IV. CONCLUSIONS AND OUTLOOK

The technological breakthroughs that are taking place drive the computing infrastructure toward many-core solutions, as clearly shown by the Top500[50] ranking and manufacturers roadmaps. The success in harnessing computing power for the needs of scientific progress depends ever more strictly on our ability to exploit these new architectures and the classic performance doubling every 18 months is in that respect no more for free: changes in programming paradigms and coding are mandatory, and it is often no simple exercise to translate transistor growth into practical performance. Our continuing optimization of the Dirac−Kohn−Sham module of the package BERTHA strives to go in that direction.

dx.doi.org/10.1021/ct400752s | J. Chem. Theory Comput. 2013, 9, 5356−5364

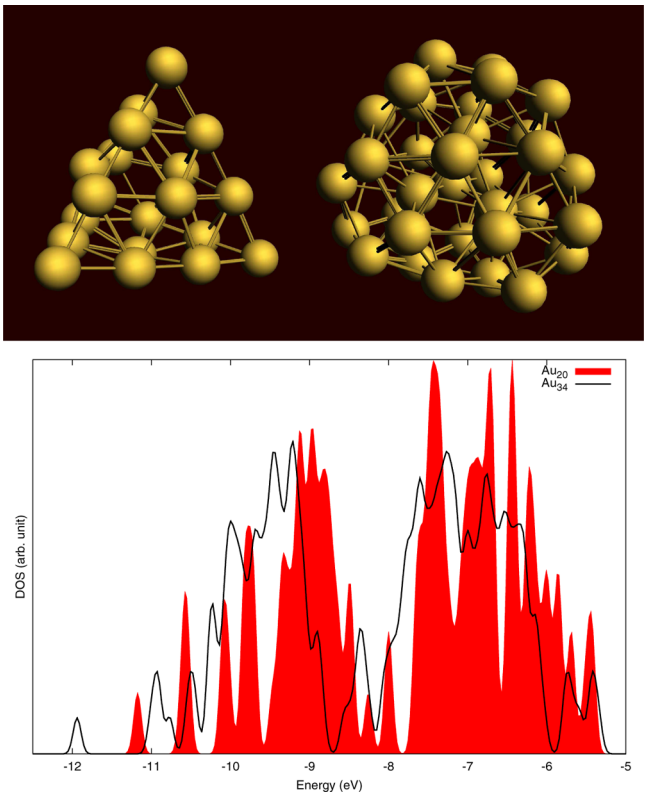


Figure 3. Density of states (DOS) of the Au_{20} and Au_{34} clusters resulting from the DKS calculations. A Gaussian broadening of 0.05 eV is used.

We completed a parallelization scheme that permits the efficient use of distributed memory. At its core is an original procedure for mapping the DKS matrix distribution between an optimal integral-driven scheme for its parallel construction, guided by the structure of our specific basis sets and by the density fitting algorithms used, and the two-dimensional block-cyclic scheme required by the ScaLAPACK routines employed to perform the required linear algebra operations. The resulting implementation of BERTHA is virtually open-ended in terms of memory requirements and extends its applicability to "arbitrarily" large systems containing heavy atoms. We showed that it can be ported very effectively on massively parallel architectures with little memory per core. The unprecedented benchmark we have reported here is an all-electron large-basis four-component relativistic DKS calculation on a Au_{34} cluster. We are now working at the effective parallelization of the remaining small serial portion of the code. The current parallelization scheme can be applied to the evaluation of exact exchange, and we are working exactly on this at the moment. We showed that DKS approach can be implemented in an efficient way and with a high numerical stability. We hope that this may stimulate new theoretical research for developing new exchange-correlation functionals depending on the four-current density.

■ ASSOCIATED CONTENT

Supporting Information

Speedup for the benchmark claculations performed on the organometallic system $[(Ph_3P)Au(C_2H_2)]^+$, as well as details about geometry, basis set, and fitting functions employed. Coordinates for the Au_{20} and Au_{34} gold clusters. This information is available free of charge via the Internet at http://pubs.acs.org.

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: loriano@storchi.org.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

This work was supported by grants from the Italian MIUR and the FIRB-futuro-in-ricerca project RBFR1022UQ. We thank the Italian SuperComputing Resource Allocation (ISCRA) through the project HP10BT1TD8 (BERTHA) for computational resources and Filippo De Angelis for enlightening discussions. L.S. would like to acknowledge IIT-SEED 2009 HELYOS and ESCORT project for financial support.

■ REFERENCES

(1) Pyykkö, P.; Desclaux, J. P. *Acc. Chem. Res.* **1979**, *12*, 276−281.
(2) Grant, I. P. *Relativistic Quantum Theory of Atoms And Molecules*, 1st ed.; Springer-Verlag, 2006; Vol. *1*
(3) Schwerdtfeger, P., Ed. *Relativistic electronic structure theory. Part 1. Fundamentals*, 1st ed.; Elsevier: Amsterdam, 2002; Vol. *1*.
(4) Reiher, M.; Wolf, A. *Relativistic Quantum Chemistry. The Fundamental Theory of Molecular Science*; Wiley-VCH: Weinheim, 2009; p 669.
(5) Autschbach, J. *J. Chem. Phys.* **2012**, *136*, 150902.
(6) Gorin, D. J.; Toste, F. D. *Nature* **2007**, *446*, 395−403.
(7) Ahuja, R.; Blomqvist, A.; Larsson, P.; Pyykkö, P.; Zaleski-Ejgierd, P. *Phys. Rev. Lett.* **2011**, *106*, 018301.
(8) Pyykkö, P. *Annu. Rev. Phys. Chem.* **2012**, *63*, 45−64.
(9) Kinoshita, T.; Fujisawa, J.-i.; Nakazaki, J.; Uchida, S.; Kubo, T.; Segawa, H. *J. Phys. Chem. Lett.* **2012**, *3*, 394−398.
(10) Scalmani, G.; Frisch, M. *J. Chem. Theory Comput.* **2012**, *8*, 2193−2196.
(11) Bulik, I. W.; Scalmani, G.; Frisch, N. J.; Scuseria, G. E. *Phys. Rev. B* **2013**, *87*, 035117.
(12) Saue, T.; Fegri, K.; Helgaker, T.; Gropen, O. *Mol. Phys.* **1997**, *91*, 937.
(13) Nakajima, T.; Hirao, K. *J. Chem. Phys.* **2004**, *121*, 3438−3445.
(14) Liu, W.; Hong, G.; Dai, D.; Li, L.; Dolg, M. *Theor. Chem. Acc.* **1997**, *96*, 75−83.
(15) Varga, S.; Engel, E.; Sepp, W.; Fricke, B. *Phys. Rev. A* **1999**, *59*, 4288−4294.
(16) Visscher, L. *Theor. Chem. Acc.* **1997**, *98*, 68.
(17) Belpassi, L.; Storchi, L.; Quiney, H. M.; Tarantelli, F. *Phys. Chem. Chem. Phys.* **2011**, *13*, 12368−12394.
(18) Komorovsky, S.; Repisky, M.; Malkina, O. L.; Malkin, V. G.; Ondik, I. M.; Kaupp, M. *J. Chem. Phys.* **2008**, *128*, 104101.
(19) Repisk, M.; Komorovsk, S.; Malkina, O. L.; Malkin, V. G. *Chem. Phys.* **2009**, *356*, 236−242.
(20) Repisk, M.; Komorovsk, S.; Malkin, E.; Malkina, O. L.; Malkin, V. G. *Chem. Phys. Lett.* **2010**, *488*, 94−97.
(21) Belpassi, L.; Tarantelli, F.; Sgamellotti, A.; Quiney, H. M. *J. Chem. Phys.* **2006**, *124*, 124104.
(22) Mintmire, J. W.; Dunlap, B. I. *Phys. Rev. A* **1982**, *25*, 88−95.
(23) Manby, F. R.; Knowles, P. J. *Phys. Rev. Lett.* **2001**, *87*, 163001.
(24) Belpassi, L.; Tarantelli, F.; Sgamellotti, A.; Quiney, H. M. *J. Chem. Phys.* **2008**, *128*, 124108.
(25) Belpassi, L.; Tarantelli, F.; Sgamellotti, A.; Quiney, H. M. *Phys. Rev. B* **2008**, *77*, 233403.
(26) Kelley, M. S.; Shiozaki, T. *J. Chem. Phys.* **2013**, *138*, 204113.
(27) NVIDIA *CUDA Programming Guide 2.0*; 2008. http://docs.nvidia.com/cuda/ (accessed October 28, 2013).
(28) Nadathur, S. *Fast Sort on CPUs, GPUs and Intel MIC Architectures*; Technical Report, Intel Labs, 2010.

(29) Adiga, A. N. An overview of the BlueGene/L Supercomputer. *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, Baltimore, MD, Nov 16−22, 2002.
(30) Storchi, L.; Belpassi, L.; Tarantelli, F.; Sgamellotti, A.; Quiney, H. M. *J. Chem. Theory Comput.* **2010**, *6*, 384−394.
(31) Silicon Graphics. *Powering the Real-time Enterprise*; White Paper 3935, 2006.
(32) MPI: A Message-Passing Interface Standard. Version 2.2. Message Passing Interface Forum. University of Tennessee, 2009; http://www.mpi-forum.org (accessed October 28, 2013).
(33) Blackford, L. S.; Choi, J.; Cleary, A.; D'Azevedo, E.; Demmel, J.; Dhillon, I.; Dongarra, J.; Hammarling, S.; Henry, G.; Petitet, A.; Stanley, K.; Walker, D.; Whaley, R. C. *ScaLAPACK Users' Guide*; Society for Industrial and Applied Mathematics: Philadelphia, PA, 1997.
(34) Quiney, H. M.; Belanzoni, P. *J. Chem. Phys.* **2002**, *117*, 5550−5563.
(35) Quiney, H. M.; Skaane, H.; Grant, I. P. *J. Phys. B: At. Mol. Opt. Phys.* **1997**, *30*, L829.
(36) Quiney, H. M.; Skaane, H.; Grant, I. P. Ab initio relativistic quantum chemistry: four-components good, two-components bad!. In *Advances in Quantum Chemistry*; Lwdin, P.-O., Ed.; Academic Press: 1998; Vol. *32*.
(37) Belpassi, L.; Storchi, L.; Tarantelli, F.; Sgamellotti, A.; Quiney, H. M. *Future Gener. Comp. Sy.* **2004**, *20*, 739−747.
(38) Belpassi, L.; Tarantelli, F.; Sgamellotti, A.; Quiney, H. M. *J. Chem. Phys.* **2005**, *122*, 184109.
(39) Dyall, K. G. *Theor. Chem. Acc.* **2004**, *112*, 403−409.
(40) Grant, I. P.; Quiney, H. M. *Phys. Rev. A* **2000**, *62*, 022508.
(41) Becke, A. D. *Phys. Rev. A* **1988**, *38*, 3098−3100.
(42) Lee, C.; Yang, W.; Parr, R. G. *Phys. Rev. B* **1988**, *37*, 785−789.
(43) Amdahl, G. Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. In *AFIPS Conference Proceedings*; AFIPS, 1967; Vol. *30*.
(44) Geimer, M.; Wolf, F.; Wylie, B. J. N.; brahm, E.; Becker, D.; Mohr, B. *Concurrency Comput.: Pract. Exp.* **2010**, *22*, 702−719.
(45) te Velde, G.; Bickelhaupt, F. M.; Baerends, E. J.; Fonseca Guerra, C.; van Gisbergen, S. J. A.; Snijders, J. G.; Ziegler, T. *J. Comput. Chem.* **2001**, *22*, 931−967.
(46) Fonseca Guerra, C.; Snijders, J. G.; te Velde, G.; Baerends, E. J. *Theor. Chem. Acc.* **1998**, *99*, 391−403.
(47) SCM. Theoretical Chemistry, Vrije Universiteit, Amsterdam, The Netherlands. *ADF User's Guide*, Release 2008.1; 2008; http://www.scm.com (accessed October 28, 2013).
(48) van Leeuwen, R.; van Lenthe, E.; Baerends, E. J.; Snijders, J. G. *J. Chem. Phys.* **1994**, *101*, 1272−1281.
(49) Wang, J.; Wang, G.; Zhao, J. *Phys. Rev. B* **2002**, *66*, 035418.
(50) Meuer, H.; Dongarra, J. E. S.; Simon, H. TOP500 Supercomputer Sites. http://www.top500.org/ (accessed October 28, 2013).