JCTC Journal of Chemical Theory and Computation

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

Loriano Storchi,[†] Leonardo Belpassi,[†] Francesco Tarantelli,*[,†] Antonio Sgamellotti,[†] and Harry M. Quiney[‡]

Dipartimento di Chimica and I.S.T.M.−C.N.R., Università di Perugia, 06123, Italy, and ARC Centre of Excellence for Coherent X-ray Science School of Physics, The University of Melbourne, Victoria, 3010, Australia

Received October 12, 2009

Abstract: We show that all-electron relativistic four-component Dirac−Kohn−Sham (DKS) computations, using G-spinor basis sets and state-of-the-art density fitting algorithms, can be efficiently parallelized and applied to large molecular systems, including large clusters of heavy atoms. The performance of the parallel implementation of the DKS module of the program BERTHA is illustrated and analyzed by some test calculations on several gold clusters up to Au_{32}, showing that calculations with more than 25 000 basis functions (i.e., DKS matrices on the order of 10 GB) are now feasible. As a first application of this novel implementation, we investigate the interaction of the atom Hg with the Au_{20} cluster.

I. Introduction

Understanding the electronic structure and properties of molecules, clusters, and nanoscale materials containing heavy atoms represents a particularly challenging task for theory and computational science because the systems of interest have typically very many electrons, and both relativistic effects and electron correlation play a crucial role in their dynamics. The most rigorous way to introduce relativity in the modeling of molecular systems is to use the four-component formalism derived from the Dirac equation. The method of choice is density functional theory (DFT) if many electrons are involved, as is the case with large metal clusters. In DFT, which is normally cast in the form of the independent-particle Kohn−Sham model, all of the exchange-correlation effects are expressed implicitly as a functional of the electron density or, more generally, of the charge-current density.[1,2] The relativistic four-component generalization of the Kohn−Sham method, usually referred to as the Dirac−Kohn−Sham (DKS) model, was introduced several years ago.[3] Several modern implementations of this theory are available,[4–7] including the one contained in our own program, BERTHA.[8–16] The full four-component DKS formalism is particularly appealing because it affords great physical clarity and represents the most rigorous way of treating explicitly and ab initio all interactions involving spin, which are today of great technological importance.

The full four-component DKS calculations have an intrinsically greater computational cost than analogous non-relativistic approaches or less rigorous quasi-relativistic approaches, mainly because of the four-component structure in the representation of the DKS equation, the complex matrix representation that usually arises as a consequence, the increased work involved in the evaluation of the electron density from the spinor amplitudes, and the intrinsically larger basis sets usually required. This greater computational cost, however, essentially involves only a larger prefactor in the scaling with respect to the number of particles or the basis set size, not a more unfavorable power law. Schemes have been devised in order to reduce the computational cost (see, e.g., refs 17–19 and references therein). A significant step forward in the effective implementation of the four-component DKS theory is based on the electron-density fitting approach that is already widely used in the nonrelativistic context. Numerical density fitting approaches based on an atomic multipolar expansion[7] and on a least-squares fit[20] have in fact been employed in the four-component

* Corresponding author e-mail: franc@thch.unipg.it.
[†] Università di Perugia.
[‡] The University of Melbourne.

10.1021/ct900539m © 2010 American Chemical Society
Published on Web 01/22/2010

relativistic domain. Recently, we have implemented the variational Coulomb fitting approach in our DKS method,[14] with further enhancements resulting from the use of the Poisson equation in the evaluation of the integrals,[15,21,22] and also from the extension of the density fitting approach to the computation of the exchange-correlation term.[16]

The above algorithmic advances have represented a leap forward of several orders of magnitude in the performance of the four-component DKS approach and have suddenly shifted the applicability bottleneck of the method toward the conventional matrix operations (DKS matrix diagonalization, basis transformations, etc.) and, especially, to the associated memory demands arising in large-system/large-basis calculations. One powerful approach to tackle these problems and push significantly further forward the applicability limit of all-electron four-component DKS is parallel computation with memory distribution. Analogous parallelization efforts of nonrelativistic DFT codes have already been described in the literature.[23,34] The purpose of the present paper is to describe the successful implementation of a comprehensive parallelization strategy for the DKS module of our code BERTHA.

In section II, we briefly review the DKS method as currently implemented in BERTHA and the computational steps making up a SCF iteration. In section III, we describe in detail the parallelization strategies adopted here. In section IV, we discuss the efficiency of the approach as results from some large all-electron test calculations performed on several gold clusters. Finally, we will present an actual chemical application of the method for the all-electron relativistic study of the electronic structure of Hg−Au_{20} with a large basis set.

II. Overview of the SCF Procedure

In this section, we will briefly review the DKS method as it is currently implemented in BERTHA. We will mainly underline its peculiarities, especially in relation to the density-fitting procedure, and summarize the steps making up a SCF iteration and their typical serial computational cost for a large case. Complete details of the formalism can be found in refs 8, 9, 11, 14–16.

In BERTHA, the large (L) and small (S) components of the spinor solutions of the DKS equation are expanded as linear combinations of Gaussian G-spinor basis functions. A peculiar and important feature of the BERTHA approach is that the density elements, $\varrho_{\mu\nu}^{TT}(\mathbf{r})$, which are the scalar products of pairs of G spinors (labeled by μ and ν, with $T = L, S$), are evaluated exactly as finite linear combinations of scalar auxiliary Hermite Gaussian-type functions (HGTF). This formulation[9,11] enables the highly efficient analytic computation of all of the required multicenter G-spinor interaction integrals.

In the current implementation of BERTHA, the computational burden of the construction of the Coulomb and exchange-correlation contributions to the DKS matrix has been greatly alleviated with the introduction[14,15] of some effective density fitting algorithms based on the Coulomb metric, which use an auxiliary set of HGTF fitting functions. The method results in a symmetric, positive-definite, linear system,

$$\mathbf{Ac} = \mathbf{v} \tag{1}$$

to be solved in order to obtain the vector of fitting coefficients, $\mathbf{c}$. The procedure involves only the calculation of two-center Coulomb repulsion integrals over the fitting basis set, $A_{ij} = \langle f_i || f_j \rangle$, and three-center integrals between the fitting functions and G-spinor density overlaps, $I_{i,\mu\nu}^{TT} = \langle f_i || \varrho_{\mu\nu}^{TT} \rangle$. The vector $\mathbf{v}$ in eq 1 is simply the projection of the electrostatic potential (due to the true density) on the fitting functions:

$$v_i = (f_i || \varrho) = \sum_{T=L,S} \sum_{\mu\nu} I_{i,\mu\nu}^{TT} D_{\mu\nu}^{TT} \tag{2}$$

where $D_{\mu\nu}^{TT}$ are the density matrix elements.[14] In our implementation, we take further advantage of a relativistic generalization of the $\mathbf{J}$-matrix algorithm[11,25,26] and an additional simplification arising from the use of sets of primitive HGTFs of common exponents and spanning all angular momenta from zero to the target value (for details, see ref 14).

The auxiliary fitted density can be directly and efficiently used for the calculation of the exchange-correlation potential,[27–29] and we have implemented this procedure in our DKS module.[16] It is based on the solution a linear system similar to the Coulomb fitting one:

$$\mathbf{Az} = \mathbf{w} \tag{3}$$

where the only additional quantity to be computed is the vector $\mathbf{w}$ representing the projection of the "fitted" exchange-correlation potential onto the auxiliary functions:

$$w_i = \int v_{xc}[\tilde{\varrho}(\mathbf{r})] f_i(\mathbf{r}) \, d\mathbf{r} \tag{4}$$

The elements of the vector $\mathbf{w}$, involving integrals over the exchange-correlation potential v_{xc}, are computed numerically by a standard cubature scheme.[8] The cost of this step tends to become negligible, scaling linearly with both the number of auxiliary functions and the number of integration grid points. In the integration procedure, we again take advantage of our particular choice of auxiliary functions: the use of primitive HGTFs that are grouped together in sets sharing the same exponent minimizes the number of exponential evaluations at each grid point. Further computational simplification arises from using the recurrence relations for Hermite polynomials in the evaluation of the angular part of the fitting functions and their derivatives.[28] Using the above combined approach, the Coulomb and exchange-correlation contributions to the DKS matrix can be formed in a single step:

$$\tilde{J}_{\mu\nu}^{TT} + \tilde{K}_{\mu\nu}^{TT} = \sum_i I_{i,\mu\nu}^{TT}(c_i + z_i) \tag{5}$$

We have found that the reduction of the computational cost afforded by the above density fitting scheme is dramatic. Besides reducing the scaling power of the method from $O(N^4)$ to $O(N^3)$, it reduces enormously the prefactor (by up to 2 orders of magnitude) without any appreciable accuracy loss.[16] The application of the method to very large systems is, however, still impeded by the substatial memory require-

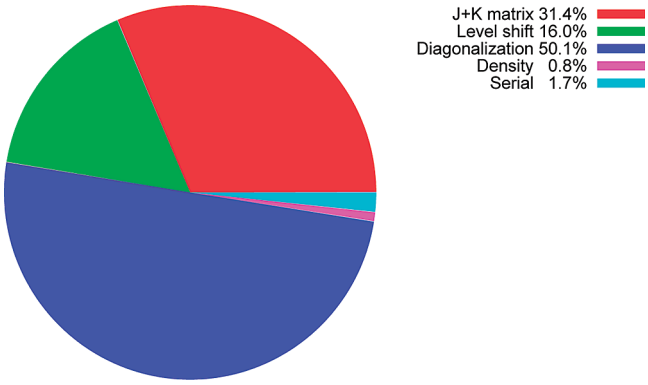

Figure 1. CPU time percentages for the various phases of a serial DKS calculation of the gold cluster Au_{16}. All linear algebra operations are performed with the Intel Math Kernel Library.

ments imposed by huge basis sets and by the consequently large matrix dimensions. A truly practical parallelization scheme must inevitably tackle these aspects of the problem.

Before we proceed to describe the parallelization of our code, it is useful to take a brief look at the time analysis of a typical serial SCF iteration of our DKS program, shown in Figure 1 for a realistically sized case: the Au_{16} cluster, with a DKS matrix dimension of 12 480. Here, we see that three $O(N^3)$ steps dominate the computation: diagonalization, $\mathbf{J} + \mathbf{K}$ matrix computation, and the level-shifting phase. Thanks to the significant progress reviewed above, the $\mathbf{J} + \mathbf{K}$ matrix computation time, formerly dominant, has been drastically reduced and, consequently, a larger time fraction (about half in the present example) is taken up by the diagonalization step. In our present serial implementation, the full DKS eigenvalue spectrum, comprising both the negative and positive energy halves, is computed. As we shall see later, we have adopted, in the parallel code, an effective reduction of the computed spectrum to the sole positive-energy occupied spinors, which affords considerable time savings. Projection methods such as those proposed by Peng et al.,[30] halving the size of the diagonalization problem, could also usefully be employed. We have not further investigated this point because, as hinted at above, our emphasis here is on the effective removal of the memory bottleneck for very large-scale applications, through data distribution. The $\mathbf{J} + \mathbf{K}$ matrix computation, which takes about a third of the time, mainly involves the calculation of the three-index integrals $I^{TT}_{i,\mu\nu}$ and the implementation of eq 5. The remaining sixth of the time is used almost entirely in the level-shifting phase, which involves the double matrix multiplication transforming the DKS matrix from basis function space to spinor space. Clearly, the parallelization effort must target these three time-consuming phases. It is remarkable that the entire density-fitting procedure, comprising the computation of the **A**, **v**, and **w** arrays and the solution of the associated linear systems of eqs 1 and 3, takes up an almost negligible fraction of the time. This phase, together with the HGTF expansion of the density, is bundled in the slice which we have labeled "Serial" in Figure 1, because we have left it unparallelized in the work described here. The remaining computation, labeled "Density", involves essentially the matrix multiplication necessary to obtain the density matrix from the occupied positive-energy spinors.

III. Parallelization Strategy

The code has been developed on a local HP Linux Cluster with an Intel Pentium D, 3.00 GHz CPU with a central memory of 4 GB on each node. The parallel implementation has been ported with success on a parallel SGI Altix 4700 (1.6 GHz Intel Itanium2 Montecito Dual Core) equipped with the SGI NUMAlink[31] interconnect. All of the results in terms of scalability and speedup reported in the following have been obtained on the latter architecture.

In the parallelization of the DKS module of BERTHA, we used the SGI implementation of the Message Passing Interface (MPI)[32] and the ScaLAPACK library.[33] The overall parallelization scheme we used can be classified as master−slave. In this approach, only the master process carries out the "Serial" portion of the SCF described in the previous section. All of the concurrent processes share the burden of the other calculation phases in Figure 1. We decided to use this approach because it is the easiest to code in order to make memory management especially convenient and favorable. Only the master process needs to allocate all of the large arrays, that is, the overlap, density, DKS, and eigenvector matrices. Each slave process allocates only some temporary small arrays when needed. In using this approach to tackle large molecular systems, it is crucial to be able to exploit the fast memory distribution scheme offered by the hardware. In particular, the SGI Altix 4700 is classified as a cc-NUMA (cache-coherent Non-uniform Memory Access) system. The SGI NUMAflex architecture[34] creates a system that can "see" up to 128 TB of globally addressable memory, using the NUMAlink[31] interconnect. The master process is thus able to allocate as much memory as it needs, regardless of the actual amount of central memory installed on each node, achieving good performances in terms of both latency and bandwidth of memory access. Some aspects of performance degradation related to nonlocal memory allocation will be pointed out later on.

A. $\mathbf{J} + \mathbf{K}$ Matrix Calculation. To parallelize the $\mathbf{J} + \mathbf{K}$ matrix construction, the most elementary and efficient approach is based on the assignment of matrix blocks computation to the available processes. The optimal integral evaluation algorithm, exploiting HGTF recurrence relations on a single process, naturally induces a matrix block structure dictated by the grouping of *G*-spinor basis functions in sets characterized by common origin and angular momentum (see also ref 12). The master process broadcasts the $\mathbf{c} + \mathbf{z}$ vector to the slaves at the outset of the computation. After this, an on-demand scheme is initiated. Each slave begins the computation of a different matrix block, while the master sets itself listening for messages. When a slave has finished computing one block, it returns it to the master and receives the sequence number identifying the next block to be computed. The master progressively fills the global DKS array with the blocks it receives from the slaves. A slave only needs to temporarily allocate the small blocks it computes.

This approach, as will be evident in the next sections, has several advantages. The communication time is essentially independent of the number of processes involved. The matrix blocks, although all relatively small, have different sizes, which tends to minimize communication conflicts, hiding communications behind computations. The small size of the blocks ensures optimal load balance and also permits a much more efficient use of the cache with respect to the serial implementation.

B. Matrix Operations. The parallelization of the matrix operations which make up the bulk of the "level shift", "diagonalization", and "density" phases, has been performed using the ScaLAPACK library routines.[33] There are two main characteristics of ScaLAPACK that we need to briefly recall here. First, the P processes of an abstract parallel computer are, in ScaLAPACK, mapped onto a $P_r \times P_c$ two-dimensional grid, or process grid, of P_r process rows and P_c process columns ($P_r \cdot P_c = P$). The shape of the grid for a given total number of processors affects ScaLAPACK performance, and we shall briefly return to this point shortly. The second fundamental characteristic of ScaLAPACK is related to the way in which all of the arrays are distributed among the processes. The input matrices of each ScaLAPACK routine must be distributed, among all of the processes, following the so-called two-dimensional block−cyclic distribution.[33] The same distribution is applied to the result arrays in output. For example, in the case of a matrix−matrix multiplication, the two input matrices must be distributed following the two-dimensional block cyclic distribution, and when the computation is done, the result matrix will be distributed among all of the processes following the same scheme.

To simplify and generalize the distribution of arrays, and the collection of the results, we first of all implemented two routines using MPI, named DISTRIBUTE(mat,rank) and COLLECT(mat,rank). The first distributes the matrix mat, originally located on process rank, among all processes, according to the block cyclic scheme; the second carries out the inverse operation, gathering on process rank the whole matrix mat block-distributed among the available processes. It is worthwhile to note that both operations, as implemented, exhibit a running time that, for a given matrix size, is very weakly dependent on the number of processes involved. Except for a larger latency overhead, this time is in fact roughly the same as that required to transfer the whole matrix between two processes. Sample timings for the distribution and collection of four double-precision complex matrices of varying sizes are shown in Table 1. The four matrices are in fact the matrices occurring in the DKS calculations of the gold clusters described later in this work.

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. Before and after the execution of these routines, we placed calls to our DISTRIBUTE and COLLECT routines to handle the relevant matrices as required. The workflow is extremely simple. Initially, the DKS matrix is distributed to all processors. After this, the "level shift", "diagonalization", and "density" steps are performed in this order, exploiting the intrinsic parallelism of the ScaLAPACK routines. At the end, we collect on the master both the density matrix and the eigenvectors. Thus, apart from the internal communication activity of the ScaLAPACK routines, there are just four explicit communication steps, namely, the initial distribution of the DKS and overlap matrices and the final gathering of the resulting eigenvectors and density matrices. Note that the only communication time in the entire calculation that depends appreciably on the number of processors involved is that of the largely insignificant initial broadcast of the $\mathbf{c} + \mathbf{z}$ vector, which is necessary to carry out the $\mathbf{J} + \mathbf{K}$ matrix construction.

Table 1. Times in Seconds for the COLLECT (C) and DISTRIBUTE (D) Routines As a Function of Matrix Size and Number of Processors

	matrix dimension							
	1560		3120		6240		12480	
number of processors	C	D	C	D	C	D	C	D
4	0.12	0.12	0.36	0.33	1.41	1.29	5.90	5.07
16	0.12	0.22	0.44	0.76	1.60	1.29	6.26	5.04
32	0.10	0.27	0.39	0.98	1.56	1.38	6.32	5.40
64	0.13	0.27	0.51	1.07	2.12	3.78	8.23	6.03
128	0.14	0.28	0.52	1.13	2.13	4.47	8.62	5.97

Before we proceed, it is necessary to add a final note about the block cyclic decomposition. The scheme is driven, besides by the topology of the processes, by the size of the blocks into which the matrix is subdivided. This size is an important parameter for the overall performance of the ScaLAPACK routines. After some preliminary tests, we chose a block dimension equal to 32 (see also refs 24 and 35), and all of the results presented here are obtained using this block dimension.

C. Notes about PZHEGVX Diagonalization. As we have seen at the end of section II, the DKS diagonalization phase takes up the largest fraction of computing time, and therefore all factors that affect its performance are important. While, for practical reasons, we could not investigate exhaustively all of these factors, we would like to highlight two of them which are particularly relevant in the present case.

The PZHEGVX routine needs some extra work arrays to carry out the diagonalization. One can execute a special preliminary dummy call to PZHEGVX in order to obtain the routine's estimate of the optimal size of this extra memory space for the case at hand. In our test applications, we noticed, however, that the estimated auxiliary memory was in fact insufficient to guarantee, especially for the larger systems, the full reorthogonalization of the eigenvectors (i.e., INFO > 0 and MOD(INFO/2,2) ≠ 0). This appeared to cause some inaccuracies and instabilities in the final results. In order to avoid these problems and also to establish common comparable conditions for all of the test cases studied, we decided to require the accurate orthogonalization of the eigenvectors in all cases. This could readily be achieved by suitably enlarging the size of the work arrays until INFO = 0. For example, in the case of the Au_{16} cluster (with a 2.4 GB DKS matrix), when 16 processors were used, the

ScaLAPACK estimate of 180 MB per processor for the work arrays had to be raised to 400 MB to achieve complete reorthogonalization. Clearly, this strict requirement is quite costly in terms of both memory and computation time, and less tight constraints might be investigated and found to be acceptable in any given practical application.

Another important aspect of ScaLAPACK diagonalization is that the PZHEGVX routine permits the selection of a subset of eigenvalues and eigenvectors to be computed. In the DKS computation, it is only essential, in order to represent the density and carry out the SCF iterations, to compute the "occupied" positive-energy spinors. This clearly introduces great savings in eigenvector computation, both in time and memory, because the size of such subset is a small fraction (about 10% in our tests) of the total. In principle, of course, selecting a subset of eigensolutions to be computed should not affect their accuracy. However, in order to ensure this numerically, so as to reproduce the results obtained by the serial code to working precision, we imposed the requirement that the computed positive-energy eigenvectors be strictly orthogonal to the ones left out. Orthogonality to the negative-energy part of the spectrum is guaranteed in practice by the very nature of the spinors and the very large energy gap that separates them. To ensure orthogonality between the occupied and virtual spinors, we found it sufficient to include a small number of lowest-lying virtuals in the computed spectrum, for which orthogonalization is explicitly carried out (as explained in the previous paragraph). We conservatively set this number of extra spinors at 10% of the number of occupied ones. As stated above, in all cases, the choice of parameters and conditions for the parallel calculations described here ensured exact reproduction of the serial results to double-precision accuracy.

D. Notes on the ScaLAPACK Grid Shape. The shape of the processor grid arrangement presented to ScaLAPACK, for a given number of processors, may affect appreciably the performance of the routines. As suggested by the ScaLAPACK Users' Manual,[33] different routines are differently influenced in this regard. The performance dependence on the grid shape is, in turn, related to the characteristics of the physical interconnection network. While an exhaustive investigation of these aspects is outside the scope of the present work, we briefly explored the effect of the grid shape on the DKS steps of BERTHA which depend on ScaLAPACK. The results are summarized in Figure 2. This essentially reports, for some of the gold-cluster calculations discussed in depth in section IV, the relative performance of the diagonalization, level-shifting, and aggregate matrix-operation steps (including the small contribution of density-matrix evaluation), observed with three different arrangements of a 16-processor array and, in the inset, a 64-processor array. In the 16-processor case, we looked at the square 4 × 4 grid shape and at the two rectangular shapes, 2 × 8 and 8 × 2, for the clusters Au_2, Au_4, Au_8, and Au_{16}. In the 64-processor setup, we examined the 8 × 8, 2 × 32, and 32 × 2 arrangements in the case of Au_{16}. The data obtained, as the figure indicates, do not allow us to draw definitive conclusions, but it is clear that the processor grid shape does indeed affect the performance of the various ScaLAPACK steps in different ways, also in dependence on the total number of processors. Lacking a general a priori model to predict the optimal grid shape in any given case and architecture, performing preliminary test calculations may be an important part of the optimization process. Both the data for $P = 16$ and for $P = 64$ seem to indicate that rectangular grids with $P_r < P_c$ perform systematically better than the reverse arrangement for which $P_r > P_c$. The former is in fact also preferable over the square $P_r = P_c$ grid for the level-shifting phase. The data for $P = 16$ suggest, however, that in the diagonalization step the square arrangement is to be preferred for small systems, while both rectangular shapes tend to become more efficient as the size of the problem increases. As a result, a $P_r < P_c$ arrangement appears to be the globally optimal choice for large computational cases. The data for $P = 64$ and Au_{16} in the inset, on the other hand, show that varying the number of processors may significantly alter the emerging pattern. In this case, for example, both rectangular grids appear again relatively unfavorable for the diagonalization step, so that the square arrangement is to be preferred overall. On the basis of the above results, we decided, for our further analysis, to use a square grid whenever possible and grids with $P_r < P_c$ otherwise.

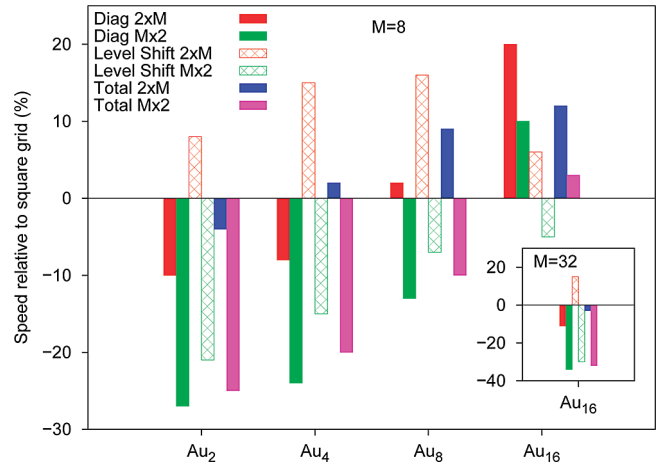

Figure 2. Performance of DKS matrix operation steps with different ScaLAPACK processor grid shapes for a series of gold-cluster calculations discussed in the present work. Shown are the data for 16 processors ($M = 8$) and for 64 processors ($M = 32$, inset, for the sole Au_{16} cluster). The histogram bars labeled "Total" refer to the sum of all three matrix operation steps, including "Density".

IV. Discussion

We performed several computations for the gold clusters Au_2, Au_4, Au_8, Au_{16}, and Au_{32}. To achieve fully comparable results throughout, we used neither integral screening techniques nor molecular symmetry in the calculations. The large component of the *G*-spinor basis set on each gold atom (22s19p12d8f) is derived by decontracting the double-ζ-quality Dyall basis set.[36] The corresponding small component basis was generated using the restricted kinetic balance relation.[11] The density functional used is the Becke 1988 exchange functional (B88)[37] plus the Lee−Yang−Parr (LYP) correlation functional[38] (BLYP). As auxiliary basis set, we use the HGTF basis called *B20*, optimized previously by us.[14]

Table 2. CPU Times (s) for the Various Phases of DKS Calculations on Some Gold Clusters as a Function of the Number of Concurrent Processes Employed (Indicated by the ScaLAPACK Grid Shape)

cluster	step	serial	2 × 2	4 × 4	4 × 8	8 × 8	8 × 16
Au_2	**J** + **K** matrix	24.75	8.63	2.43	1.69	0.90	0.90
	diagonalization	33.32	3.49	1.78	1.55	1.30	1.35
	level shift	10.20	3.59	1.14	0.71	0.58	0.51
	density	0.57	0.24	0.12	0.13	0.15	0.12
	serial	5.19	5.45	5.62	5.74	5.76	5.85
	total iteration	74.03	21.40	11.09	9.82	8.69	8.73
Au_4	**J** + **K** matrix	183.50	62.15	13.82	7.55	5.04	3.27
	diagonalization	262.07	24.87	9.02	6.86	5.32	4.82
	level shift	83.87	27.50	8.46	4.13	2.66	1.52
	density	4.48	1.55	0.72	0.56	0.66	0.49
	serial	22.01	23.26	23.72	23.65	23.72	23.16
	total iteration	555.93	139.33	55.74	42.75	37.40	33.26
Au_8	**J** + **K** matrix	1400.64	470.24	99.57	47.54	26.05	15.16
	diagonalization	1976.78	256.65	67.56	41.11	28.29	24.13
	level shift	679.52	214.64	63.44	32.16	18.88	9.56
	density	35.64	12.54	4.02	2.99	2.85	2.20
	serial	106.36	122.32	111.14	112.14	111.94	109.45
	total iteration	4198.94	1076.39	345.73	235.94	188.01	160.50
Au_{16}	**J** + **K** matrix	10946.55	3655.87	752.26	364.35	182.52	95.11
	diagonalization	17463.86	2639.84	660.06	312.19	214.92	186.94
	level shift	5598.50	2254.54	477.62	238.30	133.56	71.84
	density	293.90	91.20	25.83	19.82	13.28	9.72
	serial	580.43	598.18	597.56	599.88	597.57	596.09
	total iteration	34883.24	9239.63	2513.33	1534.54	1141.85	959.70
Au_{32}	**J** + **K** matrix		28850.31	5848.33	2851.76	1413.57	708.57
	diagonalization		28851.37	9950.42	5100.27	3197.00	1926.66
	level shift		17406.51	5471.35	2677.06	1794.60	695.77
	density		855.82	256.04	168.68	127.80	73.31
	serial		3632.59	3684.21	3729.45	3692.64	3629.83
	total iteration		79596.60	25210.35	14527.22	10225.61	7034.14

A numerical integration grid has been employed with 61 200 grid points for each gold atom. The five Au clusters chosen offer testing ground for a wide range of memory requirements and double-precision complex array handling conditions: the DKS matrix sizes were 1560 for Au_2 (37.1 MB), 3120 for Au_4 (148.5 MB), 6240 for Au_8 (594.1 MB), 12480 for Au_{16} (2.3 GB), and 24 960 for Au_{32} (9.3 GB).

Table 2 reports some elapsed times of the phases of the SCF iterations, for the various gold clusters and different numbers of processors employed (shown in their $P_r \times P_c$ ScaLAPACK arrangement). Note that the parallel diagonalization times are disproportionately smaller than, and not comparable with, the serial times because the latter involve the computation of the whole spectrum of eigensolutions, while in the parallel cases we could adopt the selection described in section III.C. It should further be noted that the Au_{32} case was too demanding to be run on a single processor and that the times shown here are averages obtained from four SCF iterations.

Figure 3 shows the corresponding speedup for the various cases. Because of the remarks just made concerning the serial diagonalization and the Au_{32} case, the speedup for the diagonalization step and for the entire Au_{32} calculation could not be computed with reference to the serial calculation. In these cases, the figure shows the relative speedup with respect to the four-processor case. This appears to be a consistent and reliable procedure. Note, for example, that the total-iteration serial time for Au_{32} estimated from the data of the 2×2 processor performance (2.79×10^5 s) agrees within 0.1% with the estimate one obtains by fitting an N^3 power law to the serial times for the smaller clusters. In Figure 3, we see that, with some exceptions, the speedup generally tends to increase with the size of the system under study. In particular, the time for the construction of the DKS matrix scales extremely well for large systems, reaching 91% and 95% of the theoretical maximum with 128 processors for Au_{16} and Au_{32}, respectively. It should be noted that the maximum speedup is one less than the number of processors in the array because of the master−slave approach used here. The other phases of the calculation scale less satisfactorily, reflecting the performance limitations of the underlying ScaLAPACK implementation. This is especially evident for the calculation of the density matrix, which is however a particularly undemanding task and a small fraction (<1%) of the whole workload. Using 32 and 64 processors, the performance of the diagonalization and level-shifting steps for the largest, Au_{32}, cluster is found to be particularly poor, appearing however to have promisingly ample room for improvement when more processors are involved. As already mentioned, the performance of the ScaLAPACK routines appears to be affected by the interplay of several contributing factors, which is not easy to unravel. Besides the array distribution block size and the shape of the processor grid, another crucial factor to consider is the global memory requirement per processor, which depends, besides application demands, on the ScaLAPACK implementation and requisites. When this exceeds the locally available memory, so that the system must resort to remote allocation through NUMAflex, or other devices in general, performance is likely to degrade, possibly quite significantly. By contrast, the diagonalization step for Au_{16} and 32 processors shows a slightly superlinear performance, which is probably related

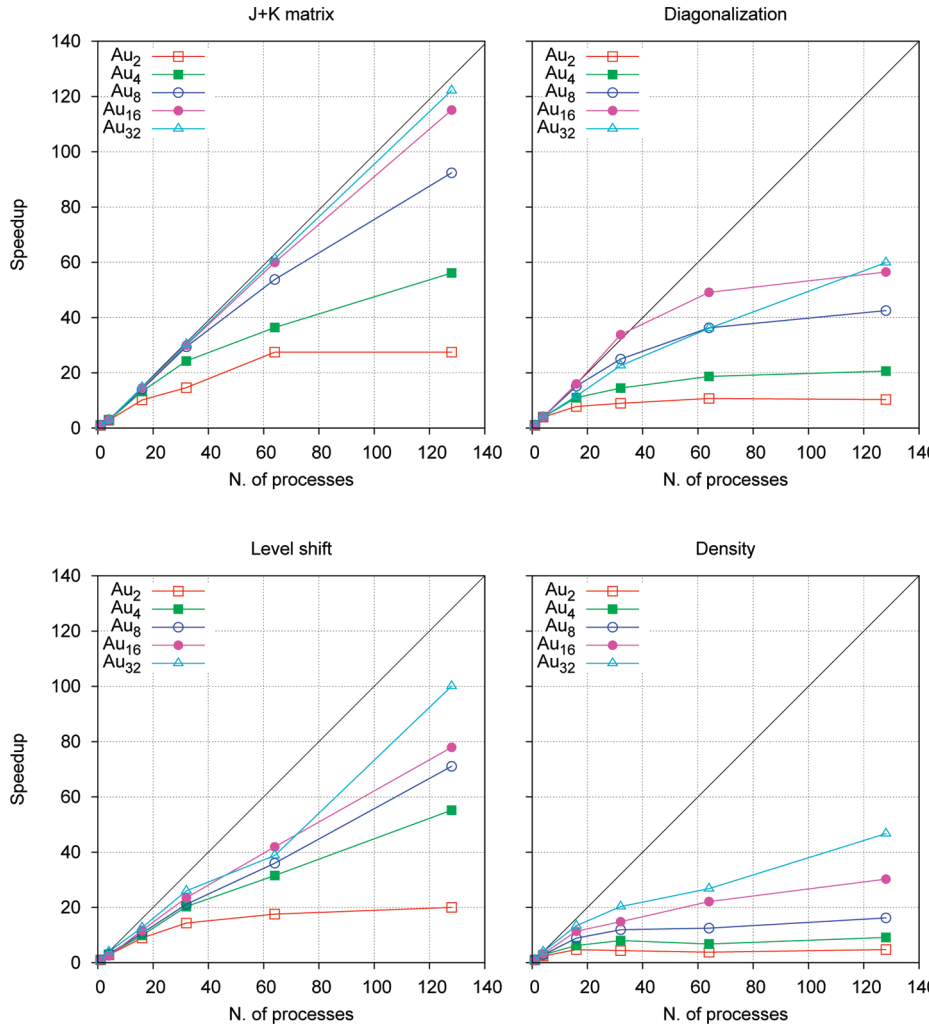

Figure 3. Speedup of the DKS computation steps for the gold clusters studied in the present work.

to the fact that the 32-processor grid is the smallest one having a nonsquare shape (see Table 2 and section III.D).

Finally, Figure 4 displays a plot of the resulting global speedup for the SCF DKS iterations. The speedup for each gold cluster is shown as the band enclosed by the measured value on the lower limit and the theoretical maximum on the upper limit. The latter is computed by taking into account the unparallelized fraction ("Serial" phase) according to Amdahl's law.[39] As can be seen, the performance for the larger clusters 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. Considered together with the great step forward represented by the memory distribution scheme, we deem this to be a very satisfactory result, clearing the way to tackle previously unfeasible systems.

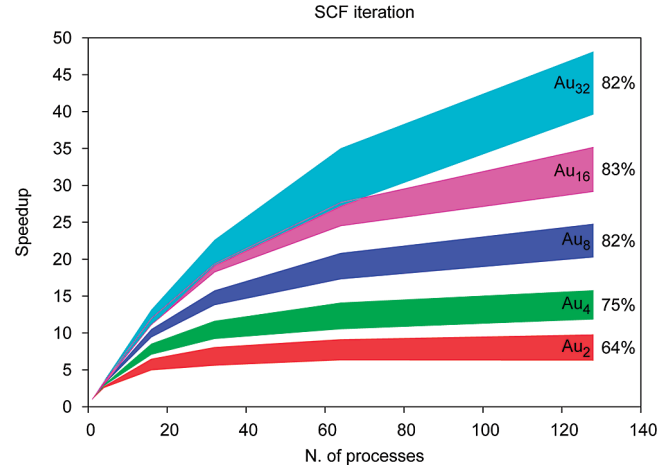

Figure 4. Overall speedup for the DKS calculation of some gold clusters. The speedup for each cluster is shown as a band delimited, on the lower side, by the measured value and, on the upper side, by the theoretical maximum (Amdahl's law). The upper limit for Au_{16} is evidenced as a line running over the Au_{32} band in the region where the two bands overlap. The measured efficiency (percentage of the upper limit) with 128 processors is shown on the right side of each band.

V. Application: Hg Atom Interacting with the Au_{20} Cluster

Understanding the nature of the interactions involving mercury and, in particular, its interaction with gold clusters and surfaces is currently of great interest and is a formidable computational task (see for instance ref 40–46 and references therein). Here, we demonstrate the practicality and effectiveness of our new parallel implementation by applying it to the characterization of the interaction of the mercury atom

with the Au_{20} cluster. In addition to the determination of some spectroscopic parameters like equilibrium bond length (R) and dissociation energy (D_e), we also perform a detailed analysis of the modification of the full relativistic all-electron density of Hg and Au_{20}, which leads to the formation of the Hg−Au_{20} bond.

A. Description of the Calculations. While the determination of the global equilibrium structure of gold clusters is itself a topic of great current interest (see for instance ref 47 and the references therein), the structure of the neutral Au_{20} cluster appears to be well established, having been successfully identified in a gas phase vibrational spectroscopy experiment combined with quantum chemical calculations.[48] These confirmed that the neutral cluster retains the symmetric pyramidal geometry established for the anion.[49] We have performed a preliminary optimization of this Au_{20} structure using the zero order relativistic approximation (ZORA) with small core and a QZ4P basis set, as implemented in the ADF package.[50–53] We then placed the Hg atom above the Au_{20} pyramid vertex, which was found to be a preferred position for an interacting noble-gas atom,[48] and further fully optimized the whole adduct using the same method. Using our parallel DKS program, as described in the previous sections, we then further reoptimized the Hg−Au internuclear distance, keeping the Au-cluster structure fixed. The interaction energy was determined by the difference in total energy of Hg−Au_{20} and the fragments Hg and Au_{20}.

The large component of the G-spinor basis set that we used was obtained by decontracting the Dyall basis set of triple-ζ quality (29s24p15d11f3g1h) on both gold and mercury atoms.[36] This is a larger basis set than that used in the Au_2−Au_{32} test calculations. The corresponding small component basis was generated using the restricted kinetic balance relation.[11] This results in a DKS matrix of dimension 24 444 (about 8.9 GB double-precision complex numbers). As the basis set for density fitting we used the set B20 described elsewhere.[14,16] This comprises 307 Hermite Gaussian functions on each heavy atom. The density functional used is the Becke 1988 exchange functional (B88)[37] plus the Lee−Yang−Parr (LYP) correlation functional[38] (BLYP). All calculations were carried out with a total energy convergence threshold of 10^{-7} hartree.

The Au−Hg bond length was determined iteratively using a quadratic fit to the energy, requiring several DKS single points. The calculations have been performed on the SGI Altix 4700 described above, using 64 processors. The time required to complete a single SCF iteration was about 1.5 h, and each single-point DKS calculation required about 20 iterations to reach convergence.

B. Results and Discussion. The Hg−Au_{20} bond length resulting from our calculations is 2.82 Å, and the corresponding interaction energy is 0.337 eV (32.5 kJ/mol). For comparison, the ZORA/QZ4P distance is 2.86 Å and the interaction energy is 27.5 kJ/mol. The complete optimized geometry of the Au_{20} cluster and that of the Au_{20}−Hg adduct are available as Supporting Information (SI). These results clearly suggest the presence of a chemically relevant interaction. A first qualitative insight into the nature of this interaction is provided by the graphical representation of the electronic density difference between the complex and the noninteracting fragments placed at the same geometry. In Figure 5, we show a 3D contour plot of such a DKS/BLYP electron density difference. A surprising feature of this is its very rich and complex structure, reaching the most remote regions of the Au cluster far away from the interaction zone. The density accumulation is particularly pronounced in the Hg−Au internuclear region and in the zone between the first two gold layers (i.e., between the gold atom bound to Hg and the three neighbors below it). A significant density depletion zone is observed instead on the far side of the Hg atom, opposite the Hg−Au bond.

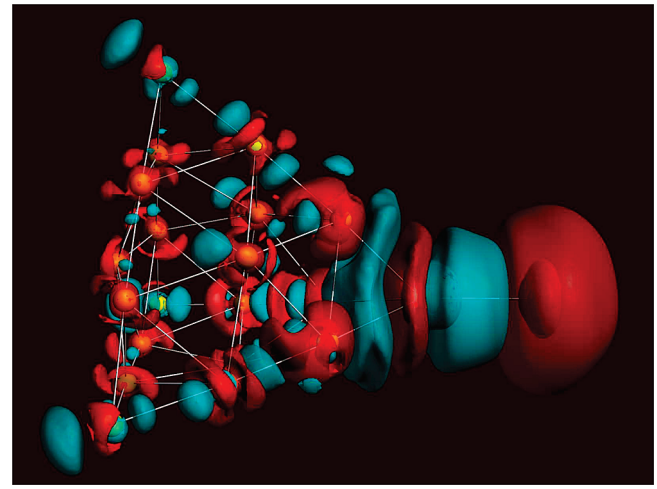

Figure 5. DKS/BLYP contour plot of the electron density difference upon bond formation between the atom Hg and the Au_{20} cluster. Red isodensity surfaces identify zones of density decrease, blue ones of density increase. The density value at the surfaces is $\pm 0.0001\,e/\text{au}^3$.

It is common to investigate the changes in charge density that result from the binding of an adsorbate to a surface by computing the density difference along the binding direction z, $\Delta\varrho(z)$.[41] This is given by

$$\Delta\varrho(z) = \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} dy\, \Delta\varrho(x, y, z) \qquad (6)$$

where $\Delta\varrho(x, y, z)$ is the electron density of the complex minus that of the two isolated fragments. $\Delta\rho(z)$ for the Hg−Au_{20} system is shown in the top panel of Figure 6. Here, the z axis is that passing through the positions of the Hg atom and the Au atom nearest to it. Positive values of the function denote accumulation of charge, while negative values indicate regions where the charge density is depleted. Inspection of this plot makes clearly more detailed the qualitative information obtained from Figure 5. Note in particular the marked density accumulation in the Hg−Au bond region and in the vicinity of the second gold layer, accompanied by density depletion around the Hg atom, especially on the far side, and to the left side of the nearest Au atom. This electron charge depletion at the Au site on the opposite side of a coordinating bond is an important feature of gold chemistry observed also in previous studies.[54,55] There are evident oscillations in the density difference around Hg and the nearest Au which may be put in relation with the structure of the electronic density of the atoms along z already observed and discussed.

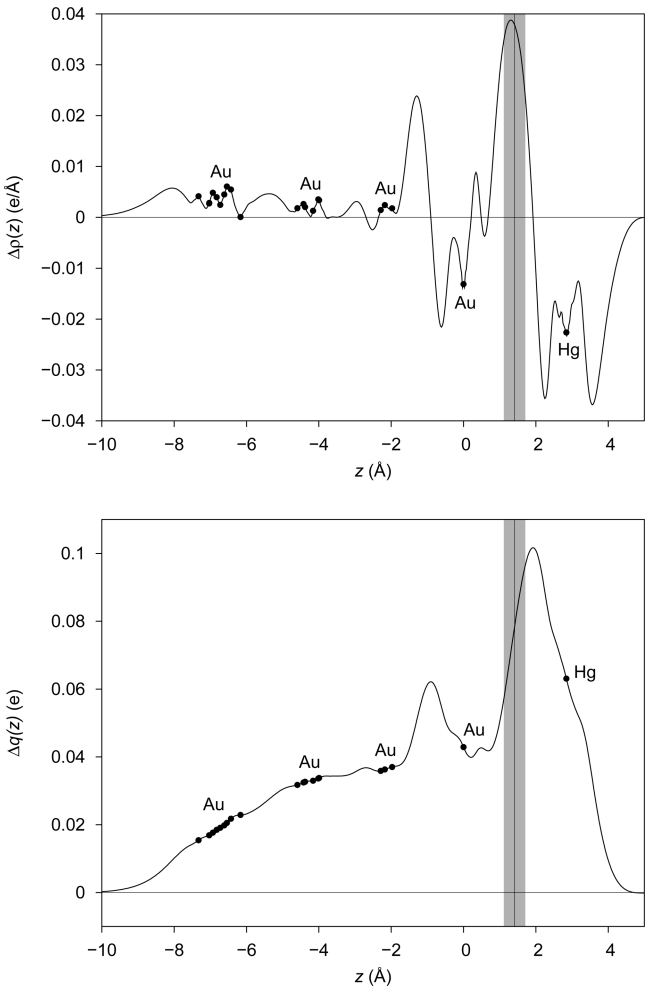

Figure 6. Upper plot: DKS electron density change along the Hg−Au bond direction [eq 6] upon formation of the $Hg-Au_{20}$ system. The circles on the curve mark the projection of the position of the indicated atoms. The vertical gray strip marks a region of width equal to 20% of the Hg−Au distance centered about the *z* position (vertical line) at which the densities of the noninteracting fragments cross (see the text for more details). Lower plot: Electron charge displacement function $\Delta q(z)$ [eq 7] for the same system.

A more immediately informative picture of the bonding may be obtained by a progressive integration of eq 6 along the internuclear axis, that is, by the function[54]

$$\Delta q(z) = \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} dy \int_{-\infty}^{z} \Delta\varrho(x, y, z')\, dz' \qquad (7)$$

This measures the actual electron displacement taking place upon bond formation, that is, the amount of electron charge transferred into the integration region up to z as one moves from left to right along the axis. In other words, $\Delta q(z)$ is the charge displaced from the right to the left side of the plane perpendicular to the axis in z. Thus, a negative value indicates a charge transfer (CT) of that magnitude from left to right, and similarly, the difference between two Δq values gives the net electron influx into the region delimited by the corresponding planes.

The plot of $\Delta q(z)$ for $Hg-Au_{20}$ is shown in the lower panel of Figure 6. The most immediately eye-catching feature of the plot is that $\Delta q(z)$ is appreciably positive everywhere in the complex region. This means that there is a shift of charge from the Hg atom toward gold which does not stop at the nearest Au layers but, surprisingly, extends appreciably even beyond the fourth layer. The Δq function shows two peaks, one corresponding to the already observed charge fluctuation between the first and second Au layers, and the second corresponding to the charge accumulation in the Hg−Au region followed by a decrease of about $0.1e$ around the Hg atom. If one defines an arbitrary boundary separating the Hg atom from the gold cluster, a corresponding effective CT value between the two fragments may be quantified. In Figure 6, we have shown one such plausible boundary, which has already been proposed in other cases,[54–56] corresponding to the point along z where equal isodensity surfaces of the noninteracting fragments become mutually tangent. Remarkably, this point almost coincides with the peak of density accumulation (maximum of $\Delta\varrho(z)$ or maximum slope of Δq). The CT value at this point is $0.08e$, which is quite close to the value $0.09e$ recently proposed by Steckel[41] for a Hg atom interacting with a Au(001) surface. Interestingly, similar CT values were found in the comparably weak covalent bond between AuF and the heavier noble gases.[54] For those adducts, a detailed comparison was carried out between DKS/BLYP and four-component coupled-cluster results concerning a vast array of molecular properties. As noted above, in spite of the weakness of the Hg−Au interaction, this transferred charge is delocalized over a surprisingly large distance in the gold cluster. The Δq curve is quite flat between the second and third gold layers, meaning that electrons do not accumulate here, so that nearly half of the total charge transferred is still found beyond the third gold layer. This insightful picture of the large spatial extent of the chemical interaction between Hg and a gold cluster represents a major novel result of the present investigation. The extent of charge delocalization found here may be overestimated by the BLYP functional self-interaction error (see for example refs 57 and 58 and references therein). It would indeed be of great interest to investigate exhaustively the behavior of different functionals with regard to the nature of similar chemisorption bonds. The computational advances documented in the present work will make such developments feasible and worthwhile in the rigorous four-component, all-electron framework. The all-electron character of the present approach is especially useful to bring to light changes in the electronic structure close to the nuclei which in general cannot be expected to be properly described by effective-core-potential or frozen-core methods. This is naturally of relevance for the calculation of properties which sensitively probe the electron density near heavy nuclei.

VI. Conclusions and Outlook

In this work, we presented an effective, essentially complete, parallelization of a relativistic all-electron four-component Dirac−Kohn−Sham program called BERTHA. We used MPI for all data communication of the parallel routines, and the ScaLAPACK library to perform the most demanding matrix operations. The main aim of the parallelization scheme adopted was the effective reduction of memory requirements per processor through array distribution, enabling the han-

dling of the DKS matrices arising in large-scale calculations on heavy-atom chemical systems using large basis sets. The scheme employs a master−slave model for the DKS matrix construction, relying on the automatic distributed memory allocation available on the SGI Altix architecture used. This procedure may be immediately transported to a generic conventional cluster provided a single node has enough memory to accommodate the arrays. It also lends itself straightforwardly, with minor changes, to explicit memory distribution by keeping track of array block location. In both cases, effective interprocess communication would even be reduced compared to the present implementation. The performance of the DKS matrix construction algorithm, including density fitting, was found to be excellent, reaching 95% of the linear limit for a large-basis Au_{32} cluster on 128 processors.

The block-cyclic array decomposition required by ScaLAPACK was handled by explicitly written routines, both for the distribute and collect operations. The overall communication for the ScaLAPACK-dominated steps of the calculation amounts to one distribution of the relevant arrays and one final collection, per SCF iteration. Using our routines, communication time was conveniently found to be largely independent of the number of processors used, as ideally expected. The performance of the ScaLAPACK steps degrades, however, somewhat compared to the DKS construction step. It further depends appreciably on the complicated and not easily foreseeable interplay of several factors, including distribution blocksize, processor grid shape, number of processors, and memory requirement per processor. We undertook a preliminary investigation of the effect of some of these variables, which may be of interest also for other chemical applications. The global performance of the DKS calculation in large test cases was however quite satisfactory, reaching over 80% of the theoretical limit dictated by Amdahl's law.

As a first chemically significant application of the new all-electron DKS parallel code, we have studied the interaction of a Hg atom with a gold cluster of 20 atoms, using a triple-ζ basis set of 24 444 functions (a DKS matrix of about 8.9 GB). A detailed study of the electronic density modification caused by the interaction shows clearly that the bond exhibits a marked covalent character and that it is characterized by a significant charge transfer, of about 0.08 electron charges in magnitude, from the mercury atom to the gold cluster. We were able to demonstrate the rather unexpected and interesting feature that the charge transferred is significantly delocalized over the entire cluster, about half of it to be found as far away from the interaction site as the third and fourth gold atom layers.

Acknowledgment. This work has been carried out as part of an Extreme Computing Initiative project of the Distributed European Infrastructure for Supercomputing Applications (DEISA). We thank I. Girotto and G. Erbacci of CINECA and the staff of Leibniz Rechenzentrum (Garching, Germany) for continuous assistance, in particular, S. H. Leong, A. Block, and M. Allalen.

Supporting Information Available: Cartesian coordinates and contour plots of the elctron density difference at the Hg site. This material is available free of charge via the Internet at http://pubs.acs.org.

References

(1) Vignale, G.; Kohn, W. *Phys. Rev. Lett.* **1996**, *77*, 2037–2040.

(2) Kümmel, S.; Kronik, L. *Rev. Mod. Phys.* **2008**, *80*, 3.

(3) MacDonald, A. H.; Vosko, S. H. J. *Phys. C: Solid State* **1979**, *12*, 2977.

(4) Yanai, T.; Iikura, H.; Nakajima, T.; Ishikawa, Y.; Hirao, K. *J. Chem. Phys.* **2001**, *115*, 8267–8273.

(5) Saue, T.; Helgaker, T. *J. Comput. Chem.* **2002**, *23*, 814–823.

(6) Varga, S.; Engel, E.; Sepp, W.-D.; Fricke, B. *Phys. Rev. A* **1999**, *59*, 4288–4294.

(7) Liu, W.; Hong, G.; Dai, D.; Li, L.; Dolg, M. *Theor. Chem. Acc.* **1997**, *96*, 75–83.

(8) Quiney, H. M.; Belanzoni, P. *J. Chem. Phys.* **2002**, *117*, 5550–5563.

(9) Quiney, H. M.; Skaane, H.; Grant, I. P. J. *Phys. B: At. Mol. Opt.* **1997**, *30*, L829.

(10) Quiney, H. M.; Skaane, H.; Grant, I. P. Ab Initio Relativistic Quantum Chemistry: Four-Components Good, Two-Components Bad! In *Advanced Quantum Chemistry*; Lwdin, P.-O., Ed.; Academic Press: New York, 1998; Vol. 32, pp 1−49.

(11) Grant, I. P.; Quiney, H. M. *Phys. Rev. A* **2000**, *62*, 022508.

(12) Belpassi, L.; Storchi, L.; Tarantelli, F.; Sgamellotti, A.; Quiney, H. M. *Future Gener. Comp. Sy.* **2004**, *20*, 739–747.

(13) Belpassi, L.; Tarantelli, F.; Sgamellotti, A.; Quiney, H. M. *J. Chem. Phys.* **2005**, *122*, 184109.

(14) Belpassi, L.; Tarantelli, F.; Sgamellotti, A.; Quiney, H. M. *J. Chem. Phys.* **2006**, *124*, 124104.

(15) Belpassi, L.; Tarantelli, F.; Sgamellotti, A.; Quiney, H. M. *J. Chem. Phys.* **2008**, *128*, 124108.

(16) Belpassi, L.; Tarantelli, F.; Sgamellotti, A.; Quiney, H. M. *Phys. Rev. B* **2008**, *77*, 233403.

(17) Iliaš, M.; Saue, T. *J. Chem. Phys.* **2007**, *126*, 064102.

(18) Liu, W.; Peng, D. *J. Chem. Phys.* **2006**, *125*, 044102.

(19) Liu, W.; Peng, D. *J. Chem. Phys.* **2009**, *131*, 031104.

(20) Varga, S.; Fricke, B.; Nakamatsu, H.; Mukoyama, T.; Anton, J.; Geschke, D.; Heitmann, A.; Engel, E.; Bastug, T. *J. Chem. Phys.* **2000**, *112*, 3499–3506.

(21) Mintmire, J. W.; Dunlap, B. I. *Phys. Rev. A* **1982**, *25*, 88–95.

(22) Manby, F. R.; Knowles, P. J. *Phys. Rev. Lett.* **2001**, *87*, 163001.

(23) Geudtner, G.; Janetzko, F.; Köster, A. M.; Vela, A.; Calaminici, P. *J. Comput. Chem.* **2006**, *27*, 483–490.

(24) Inaba, T.; Sato, F. *J. Comput. Chem.* **2007**, *28*, 984–995.

(25) Challacombe, M.; Schwegler, E.; Almlöf, J. *J. Chem. Phys.* **1996**, *104*, 4685–4698.

(26) Ahmadi, G. R.; Almlöf, J. *Chem. Phys. Lett.* **1995**, *246*, 364–370.

(27) Laikov, D. N. *Chem. Phys. Lett.* **1997**, *281*, 151–156.

(28) Köster, A. M.; Reveles, J. U.; del Campo, J. M. *J. Chem. Phys.* **2004**, *121*, 3417–3424.

(29) Birkenheuer, U.; Gordienko, A. B.; Nasluzov, V. A.; Fuchs-Rohr, M. K.; Rösch, N. *Int. J. Quantum Chem.* **2005**, *102*, 743–761.

(30) Peng, D.; Liu, W.; Xiao, Y.; Cheng, L. *J. Chem. Phys.* **2007**, *127*, 104106.

(31) Silicon Graphics, SGI NUMAlink, White Paper 3771, 2005.

(32) MPI: A Message-Passing Interface Standard, version 2.2. *Message Passing Interface Forum*; University of Tennessee: Knoxville, TN, 2009. http://www.mpi-forum.org (accessed Jan 2010).

(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) Silicon Graphics, Powering the Real-time Enterprise, White Paper 3935, 2006.

(35) Choi, J.; Demmel, J.; Dhillon, I.; Dongarra, J.; Ostrouchov, S.; Petitet, A.; Stanley, K.; Walker, D.; Whaley, R. C. *Comput. Phys. Commun.* **1996**, *97*, 1–15.

(36) Dyall, K. G. *Theor. Chem. Acc.* **2004**, *112*, 403–409.

(37) Becke, A. D. *Phys. Rev. A* **1988**, *38*, 3098–3100.

(38) Lee, C.; Yang, W.; Parr, R. G. *Phys. Rev. B* **1988**, *37*, 785–789.

(39) Amdahl, G. Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. *AFIPS Conference Proceedings*; Thompson Books: Washington, DC, 1967; Vol. 30, pp 483−485.

(40) Zaleski-Ejgierd, P.; Pyykko, P. *J. Phys. Chem. A* **2009**, *113*, 12380–12385.

(41) Steckel, J. A. *Phys. Rev. B* **2008**, *77*, 115412.

(42) Rykova, E. A.; Zaitsevskii, A.; Mosyagin, N. S.; Isaev, T. A.; Titov, A. V. *J. Chem. Phys.* **2006**, *125*, 241102.

(43) Sarpe-Tudoran, C.; Fricke, B.; Anton, J.; Persina, V. *J. Chem. Phys.* **2007**, *126*, 174702.

(44) Munro, L. J.; Johnson, J. K.; Jordan, K. D. *J. Chem. Phys.* **2001**, *114*, 5545–5551.

(45) Gaston, N.; Schwerdtfeger, P.; Saue, T.; Greif, J. *J. Chem. Phys.* **2006**, *124*, 044304.

(46) Gaston, N.; Paulus, B.; Rosciszewski, K.; Schwerdtfeger, P.; Stoll, H. *Phys. Rev. B* **2006**, *74*, 094102.

(47) Assadollahzadeh, B.; Schwerdtfeger, P. *J. Chem. Phys.* **2009**, *131*, 064306.

(48) Gruene, P.; Rayner, D. M.; Redlich, B.; van der Meer, A. F. G.; Lyon, J. T.; Meijer, G.; Fielicke, A. *Science* **2008**, *321*, 674–676.

(49) Li, J.; Li, X.; Zhai, H.-J.; Wang, L.-S. *Science* **2003**, *299*, 864–867.

(50) 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.

(51) Fonseca Guerra, C.; Snijders, J. G.; te Velde, G.; Baerends, E. J. *Theor. Chem. Acc.* **1998**, *99*, 391–403.

(52) *ADF User's Guide, Release 2008.1*; SCM, Theoretical Chemistry, Vrije Universiteit: Amsterdam, The Netherlands, 2008. http://www.scm.com (accessed Jan 2010).

(53) van Leeuwen, R.; van Lenthe, E.; Baerends, E. J.; Snijders, J. G. *J. Chem. Phys.* **1994**, *101*, 1272–1281.

(54) Belpassi, L.; Infante, I.; Tarantelli, F.; Visscher, L. *J. Am. Chem. Soc.* **2008**, *130*, 1048–1060.

(55) Salvi, N.; Belpassi, L.; Tarantelli, F. To be published.

(56) Belpassi, L.; Tarantelli, F.; Pirani, F.; Candori, P.; Cappelletti, D. *Phys. Chem. Chem. Phys.* **2009**, *11*, 9970–9975.

(57) Cohen, A. J.; Mori-Sanchez, P.; Yang, W. *Science* **2008**, *321*, 792–794.

(58) Jiménez-Hoyos, C. A.; Janesko, B. G.; Scuseria, G. E. *J. Phys. Chem. A* **2009**, *113*, 11742–11749.

CT900539M