Kokott, Sebastian; Merz, Florian; Yao, Yi; Carbogno, Christian; Rossi, Mariana; Havu, Ville; Rampp, Markus; Scheffler, Matthias; Blum, Volker

Efficient all-electron hybrid density functionals for atomistic simulations beyond 10 000 atoms

Published in:
Journal of Chemical Physics

DOI:
10.1063/5.0208103

Published: 14/07/2024

Document Version
Publisher's PDF, also known as Version of record

The Journal of Chemical Physics

RESEARCH ARTICLE | JULY 11 2024

Efficient all-electron hybrid density functionals for atomistic simulations beyond 10 000 atoms

Sebastian Kokott ; Florian Merz ; Yi Yao ; Christian Carbogno ; Mariana Rossi ; Ville Havu; Markus Rampp ; Matthias Scheffler ; Volker Blum

J. Chem. Phys. 161, 024112 (2024)
https://doi.org/10.1063/5.0208103

AIP Publishing

The Journal of Chemical Physics ARTICLE pubs.aip.org/aip/jcp

Efficient all-electron hybrid density functionals for atomistic simulations beyond 10 000 atoms

Cite as: J. Chem. Phys. 161, 024112 (2024); doi: 10.1063/5.0208103
Submitted: 13 March 2024 • Accepted: 19 June 2024 •
Published Online: 11 July 2024

Sebastian Kokott,[1,a)] Florian Merz,[2] Yi Yao,[3] Christian Carbogno,[1] Mariana Rossi,[4] Ville Havu,[5] Markus Rampp,[6] Matthias Scheffler,[1] and Volker Blum[3,7]

AFFILIATIONS

[1] The NOMAD Laboratory at the Fritz Haber Institute of the Max-Planck-Gesellschaft and IRIS Adlershof of the Humboldt-Universität zu Berlin, Berlin, Germany
[2] Lenovo HPC Innovation Center, Stuttgart, Germany
[3] Thomas Lord Department of Mechanical Engineering and Material Science, Duke University, Durham, North Carolina 27708, USA
[4] MPI for the Structure and Dynamics of Matter, Luruper Chaussee 149, 22761 Hamburg, Germany
[5] Department of Applied Physics, School of Science, Aalto University, Espoo, Finland
[6] Max Planck Computing and Data Facility, 85748 Garching, Germany
[7] Department of Chemistry, Duke University, Durham, North Carolina 27708, USA

[a)] **Author to whom correspondence should be addressed:** kokott@fhi-berlin.mpg.de

ABSTRACT

Hybrid density functional approximations (DFAs) offer compelling accuracy for *ab initio* electronic-structure simulations of molecules, nanosystems, and bulk materials, addressing some deficiencies of computationally cheaper, frequently used semilocal DFAs. However, the computational bottleneck of hybrid DFAs is the evaluation of the non-local exact exchange contribution, which is the limiting factor for the application of the method for large-scale simulations. In this work, we present a drastically optimized resolution-of-identity-based real-space implementation of the exact exchange evaluation for both non-periodic and periodic boundary conditions in the all-electron code FHI-aims, targeting high-performance central processing unit (CPU) compute clusters. The introduction of several new refined message passing interface (MPI) parallelization layers and shared memory arrays according to the MPI-3 standard were the key components of the optimization. We demonstrate significant improvements of memory and performance efficiency, scalability, and workload distribution, extending the reach of hybrid DFAs to simulation sizes beyond ten thousand atoms. In addition, we also compare the runtime performance of the PBE, HSE06, and PBE0 functionals. As a necessary byproduct of this work, other code parts in FHI-aims have been optimized as well, e.g., the computation of the Hartree potential and the evaluation of the force and stress components. We benchmark the performance and scaling of the hybrid DFA-based simulations for a broad range of chemical systems, including hybrid organic–inorganic perovskites, organic crystals, and ice crystals with up to 30 576 atoms (101 920 electrons described by 244 608 basis functions).

I. INTRODUCTION

Density functional theory (DFT) and its approximations (DFAs) have shaped the fields of computational chemistry and materials science by providing a powerful framework to investigate molecules, nanosystems, solids, and surfaces at the atomic scale. The scaling for finding the solution to the Kohn–Sham equations for (semi)local DFAs is formally $O(N^3)$, where N is a measure of the system size, when using direct eigensolvers. In practice, the actual scaling can often be reduced to $O(N^2)$ when the cache memory of modern central processing units (CPUs) is efficiently utilized, as, e.g., demonstrated by the ELPA eigensolver.[2–4] Recent developments for linear scaling DFT have driven the field to system sizes of dizzying orders of magnitudes (up to many tens of millions of atoms).[5,6] The key to achieving linear-scaling in electronic structure methods is exploiting locality since localized basis functions with finite spatial extent lead to sparsity in the density matrix. Some prominent choices for localized basis functions are numeric atom-

TABLE I. Some published large-scale hybrid DFT calculations for different methods and codes at the time of writing. The selection is restricted to simulations with three dimensional periodic boundary conditions. NAOs: numeric atom-centered orbitals, MLWF: maximal localized Wannier functions, NGWFs: Non-orthogonal generalized Wannier functions, GTOs: Gaussian-type orbitals.

Code name	System (number of atoms)	Method	References
FHI-aims	$(H_2O)_{10192}$ (30 576 atoms)	NAOs with localized resolution of identity	This work
Quantum Espresso	$(H_2O)_{512}$ (1 536 atoms)	SCDM orbitals with adaptively compressed exchange	Ref. 35
CP2K	Rubredoxin (2 825 atoms)	GPW and auxiliary density matrix methods	Ref. 36
ONETEP	Stacked polymer chains (2 000 atoms)	NGWFs and spherical waves resolution of identity	Ref. 8
BigDFT	$(H_2O)_{512}$ (1 536 atoms)	Wavelets with GPU acceleration	Ref. 37
CRYSTAL	Amorphous silica MCM-41 (4 632 atoms)	GTOs	Ref. 38

centered orbitals (NAOs),[7] non-orthogonal generalized Wannier functions,[8] polarized atomic orbitals,[6] and Gaussian functions. For all these approaches, large-scale, semilocal DFT calculations with linear scaling were successfully demonstrated.

Local and semilocal DFAs, such as the local density approximation (LDA), generalized gradient approximations (GGAs), and meta-GGAs, often face accuracy limitations in predictions of important chemical and physical properties, especially when charge transfer or localization play an important role.[9,10] To overcome these challenges and enhance the predictive capabilities of DFT, hybrid DFAs[11–13] have long been employed. For many systems, hybrid DFAs significantly improve the prediction of electronic properties, e.g., bandgaps,[14] charge localization,[15] or the description of d-orbitals.[16] The key ingredient to hybrid DFAs is mixing the (semi)local exchange of LDA, GGAs, or meta-GGAs with some fraction of non-local exact exchange (EXX). Additional flexibility is provided by treating only a certain spatial range of the Coulomb operator non-locally within the framework of the hybrid density functionals, while keeping the remainder semilocal: by introducing a range-separation function for the Coulomb potential, a variety of different functionals can be constructed, e.g., HSE06,[13,17,18] LC-wPBEh,[19] M11,[20] and ωB97.[21] Because of its smoothness, the error function is a frequent choice to divide the Coulomb potential into long- and short-range parts,

$$v(r) = \underbrace{\frac{1-\mathrm{erf}(\omega r)}{r}}_{v_{\mathrm{SR}}(r)} + \underbrace{\frac{\mathrm{erf}(\omega r)}{r}}_{v_{\mathrm{LR}}(r)}. \tag{1}$$

Here, $r = |\mathbf{r} - \mathbf{r}'|$, ω (an adjustable inverse length) is the range-separation parameter, and $v_{\mathrm{SR}}(r)$ and $v_{\mathrm{LR}}(r)$ are the short- and long-range Coulomb potential, respectively. Other physics-inspired range separation strategies are possible as well.[22] In the following, we will refer to the individual range-separated parts of the Coulomb potential in the exchange operator as Coulomb kernels. In general, we can denote the fractions of non-local full exchange and of nonlocal short-range exchange by two parameters α and β, respectively. Thus, the following contributions to the exchange energy E_x are obtained:

$$E_x(\alpha,\beta,\omega) = \alpha E_{\mathrm{EXX}} + \beta E^{\mathrm{SR}}_{\mathrm{EXX}}(\omega) + (1-\alpha)E_{\mathrm{x-DFA}} - \beta E^{\mathrm{SR}}_{\mathrm{x-DFA}}(\omega). \tag{2}$$

Here, E_{EXX} is the EXX energy using the full Coulomb potential, and $E^{\mathrm{SR}}_{\mathrm{EXX}}(\omega)$ is the short-range EXX energy. Similarly, $E_{\mathrm{x\text{-}DFA}}$ is the semilocal DFA exchange energy for the full-range Coulomb operator, and $E^{\mathrm{SR}}_{\mathrm{x-DFA}}(\omega)$ is the short-range semilocal DFA exchange energy. Using this notation, the PBE0 functional[12,23] can be recovered by choosing $\alpha = 0.25$ and $\beta = 0$ and a typical version of the HSE06 functional[13,17] benchmarked by Krukau *et al.*[18] can be obtained by setting $\alpha = 0.0$, $\beta = 0.25$, and $\omega = 0.11$ bohr^{-1}. The long-range corrected LC-ωPBEh[19] and the long-range corrected B97 functional ωB97[21] require $\alpha = 1.0$, $\beta = -1.0$, and $\omega = 0.4$ and choosing PBE or B97 as GGA functionals, respectively. In order to cover families of functionals with $\alpha \neq 0$ and $\beta \neq 0$, it can be convenient to compute two EXX matrices within a single call to a first-principles code—one matrix for each Coulomb kernel. In the following, we refer to all types of screened (long- and short-range) and unscreened (full) EXX contributions simply as EXX contributions. The difference between these different types of EXX contributions lies just in the shape of the screened or unscreened Coulomb potential (i.e., the Coulomb kernel) and the same algorithm can be employed to evaluate the exchange contribution.

Along with the increase in accuracy, hybrid DFAs typically result in significantly larger computational cost compared to semilocal DFAs due to the need to evaluate the non-local exchange operator. In fact, a naïve implementation of the electron-repulsion integrals formally scales with $O(N^4)$ with system size N. To overcome this hurdle and enable linear-scaling [$O(N)$] hybrid DFT calculations for extended systems, various strategies have been successfully employed, e.g., linear scaling incremental Fock builds,[24] the linear exchange K (LinK) approach,[25] resolution-of-identity schemes (e.g., Refs. 26–29 and references therein), auxiliary density matrix methods,[30] non-orthogonal generalized Wannier functions,[31] transformations to maximally localized Wannier functions,[32,33] and adaptive compression in a low-rank decomposition.[34] However, the computational and book-keeping overhead that incurs in such linear-scaling approaches leads to considerably higher prefactors and more complex code, typically hindering an efficient parallelization in terms of memory and computation time. Accordingly, hybrid DFT calculations are still typically considerably more costly than standard semilocal DFAs. In Table I, we present some literature examples of large-scale hybrid DFT calculations, including the codes and methods that were employed. Evidently, several codes and implementations can facilitate hybrid DFT calculations up to several thousands of atoms in size on modern HPC architectures.

This work describes recent algorithmic improvements achieved for the EXX contributions that drastically accelerate hybrid DFT calculations for large systems (non-periodic and periodic) on existing massively parallel CPU clusters, without introducing any new approximations. The approach is implemented in the all-electron code FHI-aims[27,39–42] using numeric atom-centered orbitals (NAOs) as basis functions, but the underlying techniques are general and amenable to any other code using localized orbitals for discretization. Specifically, we build on the localized resolution-of-identity (also sometimes referred to as density fitting) implementation, originally described as RI-LVL in Refs. 27 and 41, and referred to as "2015 implementation" in the following. Exploiting localization is key to achieving high performance and a low memory footprint in the evaluation of the EXX contribution. In the limit of large system sizes and for periodic systems with a bandgap, the long-range tail of the Coulomb potential Eq. (1) will be suppressed in the exchange term because of the finite range of the density matrix.[43] Thus, the EXX term becomes effectively localized. In conjunction with an appropriate choice of localized basis functions, the EXX matrix Eq. (4) becomes sparse in real space and can be evaluated at a computational cost that scales linearly with the system size. In addition, we describe further algorithmic improvements in the code regarding the evaluation of the Hartree potential, the evaluation of the Pulay force terms, and the initialization of general index arrays for periodic boundary conditions, which could otherwise become bottlenecks at certain regimes with the new hybrid-functional implementation.

For most of this paper, we will focus on the HSE06 functional, which only uses the short-range Coulomb potential $v_{\mathrm{SR}}(r)$. This is a very popular functional that provides a good balance between accuracy and computational performance (time and memory) in large-scale simulations due to the restriction of EXX exchange contributions to a smaller range. For comparison, we also show the performance of the global hybrid functional PBE0. In all cases, the solution of the generalized Kohn–Sham equations is obtained with the direct eigensolver ELPA,[2–4] version 2023.05.001. Some key examples of system types and sizes that are now attainable are shown in Fig. 1, ranging up to 30 576 atoms in size. The details of these and further benchmarks are provided in Sec. IV below.

The impact of our work for physics applications will be significant since simulations of very large, complex systems using hybrid DFAs are now affordable on typical high-performance computing resources. In cases where hybrid DFAs matter, e.g., for energy level alignments in complex structures,[44,45] the added accuracy of hybrid DFAs can be essential. One example that made use of the hybrid DFT improvements described here is a recent study addressing isolated substitutional defects and defect complexes in a layered hybrid perovskite crystal, phenethylammonium lead iodide[1] (PEPI, also included in Fig. 1). In order to eliminate any relevant interactions of defects across supercell boundaries, structure sizes up to 3383 atoms were employed, providing direct access to the spin–orbit coupled DFT-HSE06 energy band structure and associated defect energy levels. In contrast, smaller supercell models were shown to be insufficiently large, even when many hundred atoms were included since clear dispersion features of the defect states demonstrated the presence of noticeable defect–defect interactions across unit cell boundaries. Affordable simulations of systems spanning thousands of atoms using hybrid DFAs will be equally beneficial in many other scenarios where the environment of a localized defect or chemical process needs to be sufficiently large to enable realistic results, particularly when energy levels are at issue. Our development paves the way for such simulations across chemistry and materials science.

This paper is structured as follows: first, we introduce the formulas needed for the evaluation of the EXX operator. Based on them, we describe the algorithm and the improvements that have been made compared to the earlier RI-LVL EXX implementation in Ref. 41. Then, the strong and weak scaling behaviors of the new implementation are discussed. Finally, we show benchmarks of the improved implementation for a broad range of systems covering solids, surfaces, nanosystems, clusters, and molecules.

II. DESCRIPTION OF THE REAL-SPACE FORMALISM

Here, we briefly outline the notation and formalism of the real-space evaluation of the EXX operator as implemented in FHI-aims. The basic equations are those of the initial linear-scaling implementation of Levchenko *et al.*[41] Thus, we use the notation introduced in that reference and only briefly summarize the key expressions and refer to Ref. 41 for details. The formalism works for both periodic and non-periodic systems. In the following, we present the more general formulas that account for the periodic case. The non-periodic case can be recovered by considering only $\mathbf{R} = 0$, i.e., by omitting any $\mathbf{k}$ points and Bloch sums over unit cells.

In generalized Kohn–Sham theory, the $\mathbf{k}$-dependent EXX operator K, or a fraction thereof is added to the Hamiltonian. Elements of the K operator are given by

$$K^{\sigma}_{ij}(\mathbf{k}) = \sum_{\mathbf{R}} e^{i\mathbf{k}\cdot\mathbf{R}} X^{\sigma}_{ij}(\mathbf{R}), \tag{3}$$

where the Latin symbols i and j denote the NAO basis functions and σ the spin index. The vector $\mathbf{k}$ refers to a point of the Γ-centered k-grid and the sum runs over all real-space lattice vectors $\mathbf{R}$ in the Born–von Karman cell. Using the localized resolution-of-identity (RI) approach, called RI-LVL,[28] the exchange operator in real-space $X_{ij}(\mathbf{R})$ can be written as follows:

$$X^{\sigma}_{ij}(\mathbf{R}) = \sum_{k\mathbf{R}'} \sum_{\mathbf{R}''} \sum_{\mu\mathbf{Q}'} \sum_{\nu\mathbf{Q}''} C^{\mu(\mathbf{Q}')}_{ik(\mathbf{R}')} V_{\mu\nu(\mathbf{R}+\mathbf{Q}''-\mathbf{Q}')} C^{\nu(\mathbf{Q}'')}_{jl(\mathbf{R}'')} \times D^{\sigma}_{kl}\left(\mathbf{R}+\mathbf{R}''-\mathbf{R}'\right), \tag{4}$$

where $C^{\mu(\mathbf{Q}')}_{ik(\mathbf{R}')}$ are the RI expansion coefficients and

$$D^{\sigma}_{kl}(\mathbf{R}) = \frac{1}{N_{\mathbf{k}}} \sum_{\mathbf{k}} \sum_{m} f_{m\sigma}(\mathbf{k}) c^{k}_{m\sigma}(\mathbf{k}) c^{l*}_{m\sigma}(\mathbf{k}) e^{i\mathbf{k}\mathbf{R}} \tag{5}$$

is the Fourier transform of the density matrix, with the occupation numbers $f_{m\sigma}$, the Kohn–Sham eigenvectors $c^{k}_{m\sigma}(\mathbf{k})$, and the number of k-points in the Brillouin zone $N_{\mathbf{k}}$. $\mathbf{R}$ and $\mathbf{Q}$ denote lattice vectors; the sum over them is not restricted to the extent of the Born–von Karman cell, but solely by the overlap of the basis functions. The Greek symbols μ and ν are the indices of the auxiliary basis functions, as introduced next. All the basis functions and auxiliary basis

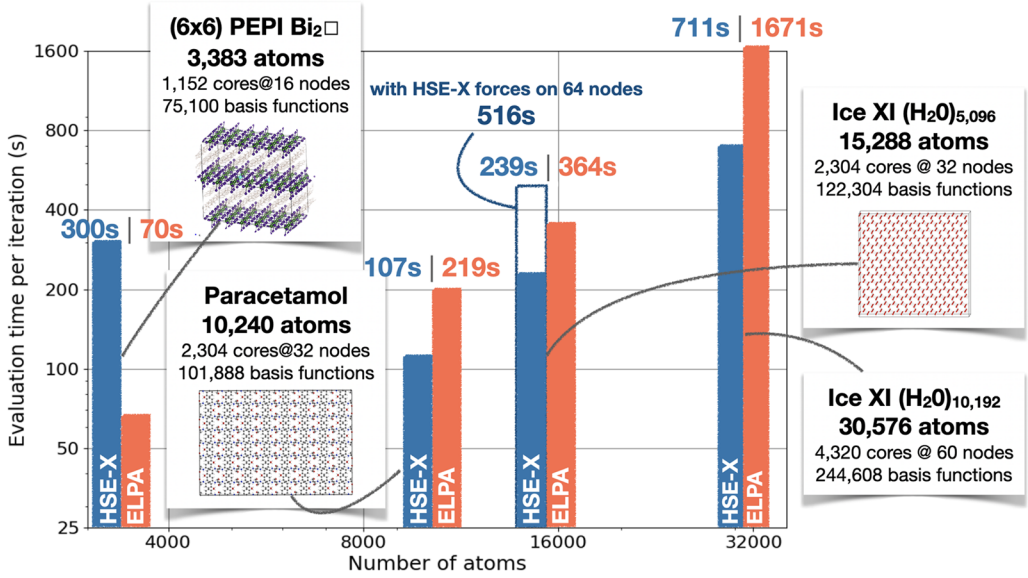

FIG. 1. Benchmark results for the largest periodic structures considered in this work. Average runtimes to evaluate the HSE06 exchange operator (blue bars) and the ELPA two-stage eigenvalue solver (red bars) per self-consistent field iteration are shown. The HSE06 hybrid functional was used for all the simulations. The following systems were simulated (from left to right): phenylethylammonium lead iodide (PEPI) with a defect complex (as indicated by the square in the chemical formula),[1] a 4 × 4 × 4 paracetamol supercell, a 15 288-atoms Ice XI supercell (including a force evaluation), and a 30 576-atom Ice XI supercell. All the calculations were carried out on the Raven HPC cluster at the MPCDF using Intel Xeon IceLake (Platinum 8360Y) nodes with 72 cores per node.

functions are labeled with a real-space lattice vector. Dropping that index refers to basis functions at an atom in the cell $\mathbf{R} = \mathbf{0}$, e.g., $i = i(\mathbf{0})$. The RI-LVL expansion is restricted in such a way that products of basis functions $\phi_i(\mathbf{r} - \mathbf{R})$ at atom I and $\phi_j(\mathbf{r})$ at atom J are expanded in terms of auxiliary basis functions $P_\mu(\mathbf{r} - \mathbf{R}')$, which must be associated with the same pair of atoms $\mathcal{P}(IJ)$,

$$\phi_i(\mathbf{r} - \mathbf{R})\phi_j(\mathbf{r} - \mathbf{R}') = \sum_{\mu(\mathbf{R})} \left(C^{\mu(\mathbf{R})}_{i(\mathbf{R})\,j(\mathbf{R}')} P_\mu(\mathbf{r} - \mathbf{R}) + C^{\mu(\mathbf{R}')}_{i(\mathbf{R})\,j(\mathbf{R}')} P_\mu(\mathbf{r} - \mathbf{R}') \right). \quad (6)$$

Formally, this leads to demanding

$$C^{\mu(\mathbf{Q})}_{i(\mathbf{R})\,j(\mathbf{R}')} = 0, \qquad \text{for } \mu \notin \mathcal{P}(IJ), \quad (7)$$

a condition that can be fulfilled as the auxiliary basis set associated with $\mathcal{P}(IJ)$ approaches completeness. As a result of translational symmetry, one basis function can always be chosen to be in the cell $\mathbf{R} = \mathbf{0}$ and so the auxiliary basis functions are only located at $\mathbf{Q} = \mathbf{0}$ or $\mathbf{R}'$. Then, the RI expansion coefficients $C^{\mu(\mathbf{Q})}_{ik(\mathbf{R})}$ can be derived as

$$C^{\mu(\mathbf{Q})}_{ij(\mathbf{R})} = \sum_{\nu(\mathbf{Q}') \in \mathcal{P}(IJ)} (ij(\mathbf{R})|\nu(\mathbf{Q}'))L^{IJ}_{\nu(\mathbf{Q}')\mu(\mathbf{Q})}, \quad (8)$$

with

$$(ij(\mathbf{R})|\nu(\mathbf{Q})) = \iint \phi_i(\mathbf{r})\phi_j(\mathbf{r} - \mathbf{R})P_\nu(\mathbf{r}' - \mathbf{Q})v(\mathbf{r} - \mathbf{r}')d\mathbf{r}d\mathbf{r}' \quad (9)$$

and the inverse Coulomb matrix $L^{IJ}_{\nu\mu} = \left(V^{IJ}_{\mu\nu}\right)^{-1}$ with

$$V_{\mu\nu}(\mathbf{r}) = \begin{cases} \iint P_\mu(\mathbf{r})P_\nu(\mathbf{r}')v(\mathbf{r} - \mathbf{r}')d\mathbf{r}d\mathbf{r}', & \text{if } \mu, \nu \in \mathcal{P}(IJ) \\ 0, & \text{otherwise,} \end{cases} \quad (10)$$

where we dropped the real-space lattice indices for simplicity and $v(\mathbf{r} - \mathbf{r}')$ is the Coulomb kernel, whose form depends on the chosen range-separation approach, i.e., full-range for Hartree–Fock exchange, short- and long-range for range-separated hybrid exchange as defined in Eq. (1).

For the actual implementation, the RI coefficients C in Eq. (4) are grouped according to which atom pairs the auxiliary basis functions belong to. Following Ref. 28, the exchange matrix for each pair of atoms $A^1_1 A^2_1$ can be written as

$$X^\sigma_{i\in A^1_1\, j\in A^2_1}(\mathbf{R}) = \sum_{A^1_2(\mathbf{R}')} \sum_{k\in A^1_2(\mathbf{R}')} \sum_{\nu\in A^2_1(\mathbf{R})} F^{\nu(\mathbf{R})}_{ik} E^{\nu\sigma}_{jk(\mathbf{R}-\mathbf{R}')} + \sum_{A^2_2(\mathbf{R}'')} \sum_{l,\nu\in A^2_2(\mathbf{R}'')} \left(2G^{\nu\sigma}_{il}(\mathbf{R} + \mathbf{R}'') + H^{\nu\sigma}_{il}(\mathbf{R} + \mathbf{R}'')\right) C^{\nu}_{lj(-\mathbf{R}'')}, \quad (11)$$

where

$$E^{\nu\sigma}_{jk(\mathbf{R})} = \sum_{l\mathbf{R}''} C^{\nu}_{jl(\mathbf{R}'')} D^{\sigma}_{kl}(\mathbf{R} + \mathbf{R}''), \quad (12a)$$

$$F^{\nu(\mathbf{R})}_{ik(\mathbf{R}')} = \sum_{\mu\in A(i)} C^{\mu}_{ik(\mathbf{R}')} V_{\mu\nu(\mathbf{R})}, \quad (12b)$$

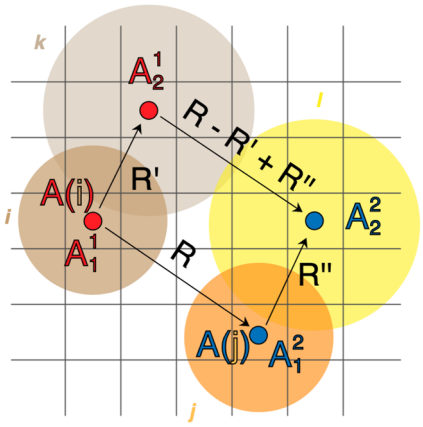

FIG. 2. Labeling of the four atom centers and the lattice vectors connecting them used for grouping the RI-LVL four-center integrals in Eq. (12). This figure is adapted from the original reference by Levchenko *et al.*[41]

$$G_{il}^{v\sigma}(\mathbf{R}) = \sum_{\mu \in A(i)} \left(E_{il(-\mathbf{R})}^{v\sigma}\right)^{*} V_{\mu v(\mathbf{R})}, \tag{12c}$$

$$H_{il}^{v\sigma}(\mathbf{R}) = \sum_{k\mathbf{R}'} F_{ki(-\mathbf{R}')}^{v(\mathbf{R}-\mathbf{R}')} D_{kl}^{\sigma}(\mathbf{R}-\mathbf{R}'). \tag{12d}$$

As shown in Fig. 2, $A(i)$ labels the atom of basis function i, $\mu \in A(i)$ indicates an auxiliary function $P_\mu(\mathbf{r})$ on atom $A(i)$. The $*$ symbol denotes complex conjugation. The choice of these intermediate matrices is motivated by the desire to minimize the number of matrix multiplications. The above-mentioned equation makes use of the translational symmetry of the RI coefficients, namely, $C_{i(\mathbf{0})k(\mathbf{R}')}^{\mu(\mathbf{R}')} = C_{k(\mathbf{0})i(-\mathbf{R}')}^{\mu(\mathbf{0})}$.

The above-mentioned expressions can be extended to also allow for the computation of derivatives, namely, the force and stress contributions stemming from Fock exchange. The details of these contributions can be found in the original publication by Knuth *et al.*[46]

The construction of the auxiliary basis set P_μ is described in Ref. 28. In brief, the auxiliary basis P_μ is constructed from the products of the original NAO basis setup to a maximum allowed angular momentum. Moreover, the auxiliary basis set can also be enhanced by additional NAO basis functions. For example, additional NAO basis functions are included in the FHI-aims species defaults, e.g., for the oxygen atom, an additional 5g function is included in the *intermediate* species defaults. In principle, no system-specific parameters are needed. Ihrig and co-workers showed that the localized expansion scheme RI-LVL to compute the Fock matrix Eq. shows significant errors in the total energy per atom (in the range of 1 meV per atom) comparing to the RI-V scheme for small auxiliary basis sets that are intended for RI-V, but the error can be systematically reduced to become negligible by using additional auxiliary basis functions. Thus, empirically we find that, for most cases, the intermediate species defaults settings give reliable results for many important observables. As a challenging example, we compare the convergence of the binding energy for the ice XI primitive unit cell (12 atoms, geometry in supplementary material, Sec. II J) for the light, intermediate, and really tight species defaults. The binding energy is given as the difference of the total energy per molecule of the crystal and the total energy of the (isolated) molecule. We use the HSE06 functional with the vdw-TS dispersion correction.[47] Using light and intermediate species defaults, we find a binding energy of −0.7010 and −0.6813 eV, respectively. As reference value for a well-converged calculation, we use the really tight species defaults and obtain a binding energy of −0.6516 eV. This converged value agrees well with the values reported in the literature, e.g., −0.6831 eV for HF+RPA.[48] Although the light and intermediate species defaults seem to give a similar result for the binding energy, in practice the intermediate species defaults are beneficial for performing high quality relaxations because a denser integration grid is used.

III. DESCRIPTION OF THE ALGORITHM AND ITS IMPROVEMENTS

A. General concepts

FHI-aims purely relies on the message passing interface (MPI) standard for parallelization. This choice has the advantage that any code implemented in this way will immediately work in parallel across multiple compute nodes, leaving the details of intra- vs cross-node communication of data arrays up to the underlying MPI library. In recent years, however, the number of CPU cores per compute node increased faster than the total available memory per node. Therefore, arrays that are needed on all MPI tasks can significantly increase memory consumption on such architectures. To address this issue, a key change in our implementation was to move any large, precomputed coefficient arrays, e.g., the RI coefficients C_{ij}^{μ} defined in Eq. (8) and the Coulomb matrix $V_{\mu v}(\mathbf{r})$ defined in Eq. (10), which are needed by all MPI tasks, to shared memory arrays that are managed according to the MPI-3 standard (i.e., by the MPI library itself). This choice improves scalability and reduces memory consumption significantly since only one copy per node instead of one copy per core is stored in the memory. For example, on a two-socket system with Intel Xeon IceLake-SP Platinum 8360Y CPUs (36 CPUs per socket, i.e. 72 CPUs per node), a reduction in the memory consumption for those arrays by roughly two orders of magnitude is achieved due this strategy alone.

Furthermore, one can exploit the fact that not all atom pairs have a significant overlap of basis functions, especially for large systems. Thus, a large number of computations and the associated memory cost can be avoided from the start. However, exploiting this sparsity requires a considerable bookkeeping effort to efficiently store, exchange, and use the sparsified data. In our optimized implementation of the EXX matrix computation, we store the global book-keeping data for the various sparse arrays in MPI-3 shared memory arrays. By this means, all MPI tasks have access to the complete metadata (MPI task and offset) and can hence access the initialized arrays and the computed results via one-sided MPI calls.

One additional advantage of the described code infrastructure is that it facilitates efficient data reshuffling. We repeatedly exploit this property to optimize the data layout for the different stages of

computation. The initialization of the Coulomb and overlap matrices, the density computation, the actual EXX matrix computation, and the Pulay mixing and storage are all performed with a different data distribution to speed up computations and to reduce load imbalance at the various stages.

Moreover, this infrastructure allows introducing an additional parallelization layer. In the 2015 implementation, the computation was only parallelized across atom pairs and the number of unit cells in the Born–von Karman cell, i.e., the set of real-space unit cells within which the Bloch phases of a finite, Γ-point centered **k**-space grid are not yet periodically repeated. This restricted, coarse-grained parallelization inherently limits the scaling for large core counts because of the amount of data that needs to be exchanged. By decoupling the data layout from the actual work, the computations of the different j columns of the exchange matrix can now be performed independently, even if all of them require the data computed during initialization. Depending on the available memory, we evenly split the global MPI communicator into n identical subcommunicators. We refer to these subcommunicators as *instances* in the following. All of those instances are set up with everything that is necessary to compute any column of the exchange matrix Eq. (11), i.e., all of them have access to the precomputed data (e.g. RI coefficients and Coulomb matrix), the communicators for parallelization across atom pairs, and temporary arrays. This allows for the computation of different chunks of the exchange matrix $X_{ij}^{\sigma}(\mathbf{R})$ [Eq. (11)] (called *blocks* in the following) to be performed independently by different instances. The gathering of all blocks of the exchange matrix at the end is very fast compared to the computation. This additional parallelization layer drastically improves scalability: when enough memory is available, multiple instances can be spawned, and the strong scaling behavior is significantly improved, as shown in the following.

In Fig. 3, we show a sketch of the new workflow for four compute nodes and a situation in which two instances are opened and three EXX matrix blocks are assigned to each of the two instances. It should be noted that this distribution can change along the self-consistent field (SCF) convergence to adjust the load balance. The number of EXX matrix blocks is given by the number of basis functions n_{basis}/block size. We explain the individual steps in more detail in Subsection III B and III C.

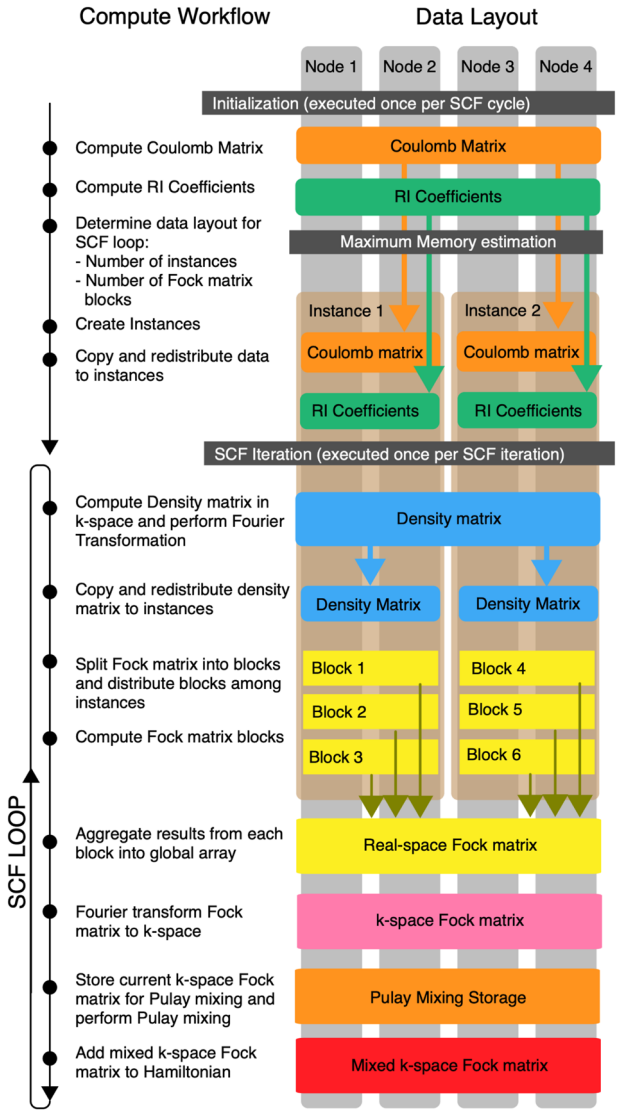

FIG. 3. Computational workflow and data layout for the Fock matrix computation for the optimized algorithm. The data layout is shown here for a calculation that uses four nodes, two instances of the computational infrastructure (see the text), and three blocks of the EXX matrix are treated by each instance.

B. Initialization

The compute workflow and a schematic data layout for the initialization are sketched in the upper half of Fig. 3. At the start of any calculation, the Coulomb matrix $V_{\mu\nu}(\mathbf{r})$ as defined in Eq. (10) and the RI coefficients C_{ij}^{μ} as defined in Eq. (8) are computed. Compared to the 2015 version, data re-use, memory access patterns, and vectorization have been improved. The parallelization for the initialization routines is updated so as to minimize load imbalance. As mentioned above, the arrays are then later redistributed and copied onto each instance to match the data layout of the computation of the EXX matrix in Eq. (4) during each SCF iteration, as indicated by the orange and green arrows shown in Fig. 3. Furthermore, the usage of data compression was extended: In the 2015 implementation, only the Coulomb matrix, Eq. (10) was compressed by removing those columns and rows that exclusively feature elements with absolute values below a threshold of 10^{-10}. The same compression method is now also used for the RI coefficients, Eq. (8). The threshold was carefully tested so as to not alter the result to a numerically significant degree for both the Coulomb matrix and the RI coefficients. Other sparsity criteria were not altered compared to the 2015 version and are explained in detail in Ref. 41, p. 65. The number of instances for the main computation is also determined during initialization. To this end, the available memory per node is measured and compared to the estimated memory consumption per instance to avoid out-of-memory situations. The memory consumption per instance is estimated based on the size of the Coulomb matrix and RI coefficients arrays, as well as heuristics for the largest temporary arrays during the main loop as defined by Eq. (12). We use the following

formula to estimate the ideal number of instances, i.e., the number that gives the best performance and still fits into memory,

$$n_{\text{instances}} = \frac{n_{\text{nodes}} \times \min\left(M_{\text{free}} - M_{\text{buffer}}\right)}{M_{\text{Coulomb}} + M_{\text{RI}} + M_{\text{main}}}, \tag{13}$$

with the total number of nodes n_{nodes}, the currently available memory per node M_{free}, the estimated memory usage needed for the actual evaluation per node M_{main}, the memory buffer per node M_{buffer}, the memory for the Coulomb matrix M_{Coulomb}, and the memory for the RI coefficients M_{RI}. However, only divisors of the total number of nodes are allowed (e.g. for four nodes only one, two, or four instances can be created), or more than one instance per node is also possible. With this mechanism, the code can use the entire available system memory over a wide range of nodes. In a strong-scaling scenario, the increasing total memory available will lead to an increasing number of instances, leading to a virtually perfect scaling of the Fock matrix computation, as long as there are enough blocks to distribute. When the memory requirements of the expected computation are high compared to the available memory, only one instance spanning all MPI tasks will be created. By this means, data redundancy is avoided and the memory is used as efficiently as possible. Effectively, this recovers the original parallelization scheme used in the 2015 version, but the remaining computational benefits of the more fine-grained load balancing and of the MPI-3 shared memory arrays still lead to a significant performance improvement. Eventually, the Coulomb matrix and RI coefficients are copied to and redistributed on each instance to achieve optimal performance during the main loop of the calculation.

As discussed above, the computation of the real-space EXX exchange matrix, Eq. (4), is performed in blocks over the last index j, which improves cache usage and reduces the number of MPI calls within an instance. We refer to this blocking of the Fock matrix as Fock matrix blocks or simply blocks in the following. The optimal block size is determined during initialization but can also be set manually as an input parameter. Since many of the temporary arrays scale with the block size, this quantity has a considerable impact on the memory consumption per instance. During initialization, the blocks of the exact exchange matrix are distributed evenly across all instances. During the main computation, the number of blocks per instance is increased/lowered after each SCF cycle according to the actual runtimes for the individual blocks to achieve optimal load balance.

In the remainder of this paper, we will collectively refer to the determination of number of instances, the determination of the Fock matrix block size, and the redistribution of the Fock matrix blocks as auto-tuning mechanisms.

C. Evaluation of the EXX matrix in real space

The EXX matrix is evaluated once per SCF iteration. The process is outlined in the lower half of Fig. 3, where the left column describes the individual steps that are executed and the right column indicates the data layout used for the largest arrays. As a first step, the (un-mixed) density matrix is constructed from the eigenvectors of the previous solution of the KS eigenvalue problem and is Fourier transformed into real space to obtain $D^{\sigma}_{kl}(\mathbf{R})$ according to Eq. (5). For the first SCF iteration, the solution of the KS eigenvalue problem for the semilocal PBE functional with an initial density of superimposed spherical free atoms or ions is used, although more sophisticated choices could be pursued in a future work. For all the subsequent SCF iterations, the density matrix from the solution of the KS eigenvalue problem using the actual hybrid functional is used. The data layout and communication patterns for the Fourier transforms in Eqs. (3) and (5) have been optimized for different stages in the computation workflow: e.g., the computation and Fourier transformation of the distributed density matrix is first computed efficiently across all MPI tasks and, for the subsequent computation steps, the real-space density matrix $D^{\sigma}_{kl}(\mathbf{R})$ is redistributed and stored in a different data layout to be optimal for the matrix multiplications in Eqs. (12a) and (12d), as indicated by the blue arrows shown in Fig. 3.

In the following, we briefly outline the computation of a row of the EXX matrix Eq. (4) in pseudocode. The notation for the temporary matrices is the same as introduced in Eq. (12). `C` refers to the RI coefficients in Eq. (8), while `C'` refers to the reordered RI coefficients to compute Eq. (12c) efficiently. `D` refers to the density matrix in real space in Eq. (5). The variable names used on the left hand side of the pseudo-instructions reflect their naming in the actual code. The workflow and the data layout that is used during the simulation is outlined in Fig. 3.

```
compute RI coefficients * density matrix:
    tmp             := E  = C*D
sum up first part of EXX matrix:
    temp_prod       := F  = C*V
    fock_matrix_mem := X += temp_prod*tmp
compute remaining temporary arrays:
    tmp2            := H  = F*D
    tmpx2           := E' = C'*D
    temp_prod       := G  = tmpx2*V
    tmp2            += 2*temp_prod
compute second part of EXX matrix:
    fock_matrix_row += C*tmp2
    fock_matrix_row += fock_matrix_mem
```

We refer to a collection of several EXX matrix rows as a *block* in the following, as also shown by the yellow boxes shown in Fig. 3. During the first SCF step, all blocks are distributed evenly among the instances, but they might be later redistributed to optimize the load balance. In order to use the available memory efficiently, several temporary arrays are re-used (and over-written) during the computation of a block. Whether or not temporary arrays or a product of temporary arrays are computed is decided based on estimates of the maximum norms of some of the arrays involved, e.g., the density matrix or the RI coefficients. These screening mechanisms have not changed and are described in detail in the 2015 paper.[41]

For systems in which not all atom pairs overlap, a dynamic load imbalance occurs in the main loop running over the global basis function index `n_basis` due to the parallelization of the Coulomb/overlap matrices over `n_atoms`. This means that the relative computational load for different MPI tasks varies between SCF iterations because the work required for a particular basis function with index `i_basis` depends on the overlap between the atom associated with `i_basis` and the atom associated with the data stored locally on the task and on the synchronization points (global

collectives) in each iteration. We implemented blocking to reduce the number of synchronization points. It should be noted that the indices `i_basis` accessed in each block are not consecutive, i.e., not associated with the same atom, but aim to achieve some balancing between all the MPI tasks within each block.

After the EXX matrix is fully constructed in real space, it is backtransformed into reciprocal space (pink box shown in Fig. 3). The Pulay mixing algorithm[49] with mixing factors based on the density is used as a default to achieve efficient SCF convergence. As described initially, we use the unmixed density matrix to construct the Fock matrix. Subsequently, we apply the Pulay mixing factors of the density to the EXX exchange matrix as well. This choice significantly increases the memory consumption for large-scale systems since, by default, the previous eight EXX matrices are stored and all of them are of the size of the Hamiltonian: $n_{\text{basis}} \times n_{\text{basis}}$ for each $\mathbf{k}$ point (orange box shown in Fig. 3). We only store EXX matrices for the set of non-equivalent $\mathbf{k}$-points per time-reversal symmetry, unless otherwise requested by the user. This saves a factor of about two for dense k-grids. In addition, the data layout has been optimized for storing the current EXX matrix while executing the Fourier back transform. Eventually, the corresponding fraction of the Pulay mixed k-space EXX matrix is added to the Hamiltonian.

D. EXX force and stress contributions

FHI-aims also provides analytical forces and stress evaluation for hybrid DFAs,[46] which enable, e.g., (periodic) structure optimization or molecular dynamics. For the theoretical background, we refer to the original paper.[46] The Fock exchange force contribution is only computed once per SCF cycle in an additional SCF step when the electronic convergence criteria have been reached. Each evaluation of the three force components and each evaluation of the six (or nine) stress tensor components require a computation of the same size and complexity as the original Fock matrix. In a naïve implementation of these Fock derivatives, the total memory consumption and runtime would increase by a factor of four, if only forces are computed or by a factor of ten if the stress is also computed. Especially for large systems, this factor would lead to huge and undesirable memory consumption. Therefore, we implemented a splitting of the computation of the stress components into several parts. The code determines the mode based on the available remaining memory. If the problem size is small, all force and stress components are computed in parallel. If the problem size is large, the force and stress computation is split into three parts: first, the exchange matrix plus the three force components are computed; second, the first three stress components; and in a third part, the remaining three stress components are computed. This choice keeps the memory consumption manageable; however, it increases the computation time by a factor of two.

E. Improvements of other relevant code parts

In addition to the EXX matrix evaluation, the evaluation of the electrostatic (Hartree) potential and of the Pulay force terms was also optimized to reduce the computational time spent for large periodic systems. This improvement benefits both semilocal and hybrid DFT computations. In FHI-aims, the Hartree potential in each SCF step is evaluated as a difference to the sum of free atom potentials, $\delta v_{\text{es}}(\mathbf{r})$.[39] The difference is computed on each point of the integration grid by summing up atom centered multipole components $\delta\tilde{v}_{\text{at},lm}(r)$ for each atom; see Ref. 39. In our implementation, we restructured the computations such that for each multipole component, its contribution to the Hartree potential is evaluated for a batch of points. This avoids branching in the innermost loop and reduces the number of subroutine calls by two orders of magnitude, improves memory accesses and cache reuse and allows for compiler vectorization.

For periodic systems, the Hartree potential is evaluated using the Ewald method that decomposes the potential into short- and long-range components. For the long-range components, we introduced a blocking method that evaluates the potential for a batch of points. Here, it was possible to rewrite the computations using highly tuned dgemm/zgemm routines of the BLAS (Basic Linear Algebra Subprograms) standard library instead of a loop-based implementation, which greatly improved the computational efficiency.

Similar to the evaluation of the Hartree Potential, we simplified the branching inside of the main loops of the Pulay force evaluation and restructured the computation to aggregate/avoid unnecessary computations. In addition, we improved the repeated initialization of some large arrays in one of the main loops by exploiting their sparsity, which also helped to reduce computation time significantly.

IV. BENCHMARK RESULTS

A. Benchmark platform (hardware and software specification)

The benchmark calculations reported here were performed on the Lenovo HPC system, *Raven*,[50] at the Max Planck Computing and Data Facility. The compute nodes provide two Intel Xeon Platinum 8360Y (IceLake-SP) processors with 72 cores per node, operated at the nominal frequency of 2.4 GHz (turbo mode disabled). Depending on the memory (RAM) requirements, we either use nodes equipped with 256 GB RAM or 512 GB RAM. Both types of nodes share the same memory-performance characteristics (~310 GB/s sustained bandwidth on the stream triad microbenchmark[51]). All the nodes are connected through a 100 Gb/s Nvidia Mellanox HDR100 InfiniBand network with a non-blocking fat-tree topology. We use the Intel ifort compiler 2021.6.0, the Intel MPI library 2021.6, and the Intel MKL library 2022.1.

B. Strong and weak scaling behavior

In the following, we compare the parallel scaling behavior of the new implementation to the original version from 2015. We show the $O(N)$ scaling with respect to increasing the number of atoms in the system, while keeping the number of basis functions per atom constant.

We compare the performance of the improved code to the original implementation from 2015[41] for GaAs supercells of different sizes. We use the $4 \times 4 \times 4$ primitive unit cell as 128-atom supercell

and subsequently double the cell in each direction, so we get supercell sizes of 256, 512, and 1024 atoms. The k-grid for the 1024-atoms supercell is chosen as $1 \times 1 \times 1$, and the k-grid density is kept consistent for the remaining supercells. We use the *intermediate 2020* species defaults of FHI-aims (which are the earlier "tight" settings in the 2010 notation as used in the 2015 implementation). The calculations are all-electron calculations with no shape approximations to the underlying electron–nuclear potential, and they are carried out without making use of any space group symmetries or other system-specific simplifications.

1. Strong scaling

The drastic improvements in the code performance are showcased in Fig. 4, which presents the actual runtimes for one evaluation of the EXX (in the original and the improved implementation) for each GaAs supercell as a function of the number of atoms included in the supercell. Regardless of the number of nodes used, a huge reduction in the runtime—roughly at least an order of magnitude—is observed for all system sizes considered. We observe a dependence of the runtime that approximately scales as aN^b with system size N. This scaling reveals that the increase in performance is rooted in both a massive reduction of the prefactor a and in improvements of the scaling exponent b. For small system sizes, the observed reduction in the prefactor by about a factor of 6 (4 nodes) and 9 (16 nodes) is largely responsible for the obtained savings. For larger system sizes, the reduced scaling exponent becomes more important: Compared to the original implementation, which already features a favorable, almost linear scaling with $b \approx 1.1$–1.26, the improvements resulted in a further decrease in the scaling coefficient, leading to an almost perfect linear scaling ($b \approx 1.01$–1.05) of the computational cost with N. This is remarkable since such a consistent linear scaling is extremely hard to achieve in actual implementations, and this is already observed for the relatively moderate system sizes considered in this plot ($N \leq 1024$ atoms).

The observed increase in the code performance also translates into a better strong scaling behavior, as shown in Fig. 5. This plot shows actual runtimes for one evaluation of the exact-exchange operator as a function of the number of nodes viz. cores. More specifically, GaAs supercells with 256, 512, and 1024 atoms were investigated using both the original implementation and the one including our improvements. As mentioned when discussing Fig. 4, we already observe large improvements in the runtime of roughly one order of magnitude for small node counts. The improvements are even more pronounced for larger-scale calculations featuring thousands of cores and can reach reductions in the runtime by two orders of magnitude and even more. Compared to the original implementation, in which the runtime $t(n) \approx 1/n^c$ scales with respect to the number of cores, n, with an exponent c between 0.2 and 0.25, the improved implementation scales with an exponent c between 0.9 and 0.95, very close to an ideal speedup. Clearly, a more favorable scaling is observed for larger workloads viz. system sizes. As shown in the next section, the developed routines show a similarly good scaling on a much larger number of nodes and cores if the respective problem size is increased accordingly.

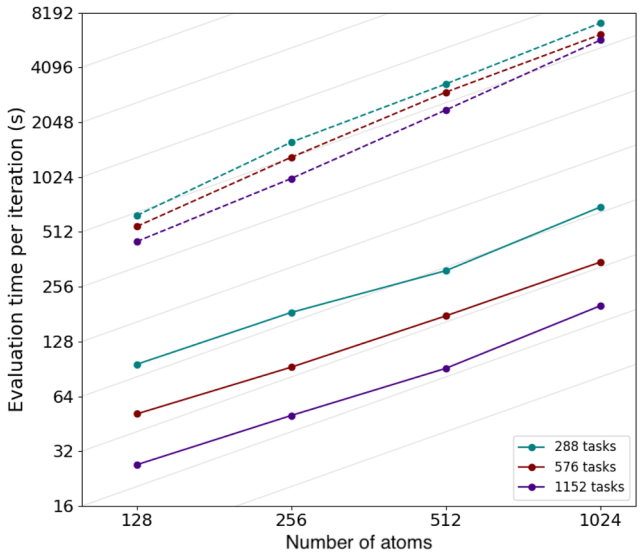

FIG. 4. O(N) scaling of the improved (solid lines) and the 2015 (dashed lines) implementation of the HSE06 exchange evaluation timings per SCF iteration for GaAs supercells with 256, 512, and 1024 atoms using the *intermediate* FHI-aims species defaults, i.e., 34 NAO basis functions per atom. The gray lines indicate linear scaling. The calculations were run on nodes with 256 GB RAM.

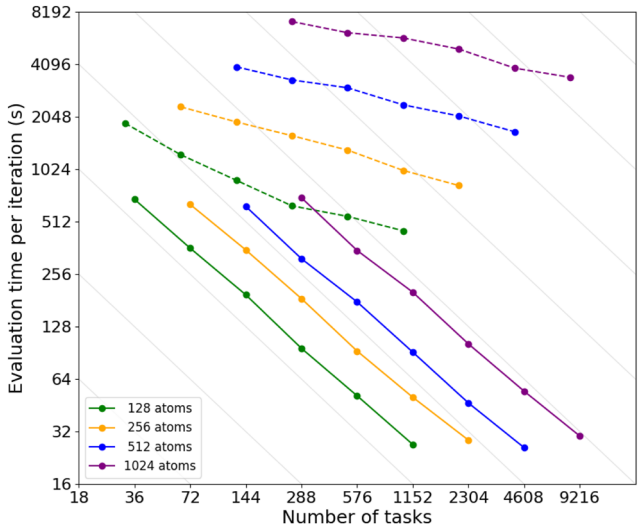

FIG. 5. Strong scaling of the improved (solid lines) and the 2015 (dashed lines) implementation of the HSE06 exchange evaluation timings per SCF iteration for GaAs supercells with 256, 512, and 1024 atoms using the *intermediate* FHI-aims species defaults, i.e., 34 NAO basis functions per atom. The gray lines indicate ideal strong scaling. The calculations were run on nodes with 256 GB RAM.

2. Weak scaling

To showcase weak scaling with a constant workload per node, cf. Fig. 6, we use the same GaAs supercell data as in the previous section. For the 2015 implementation, we observe that the average run time for a constant workload increases almost linearly

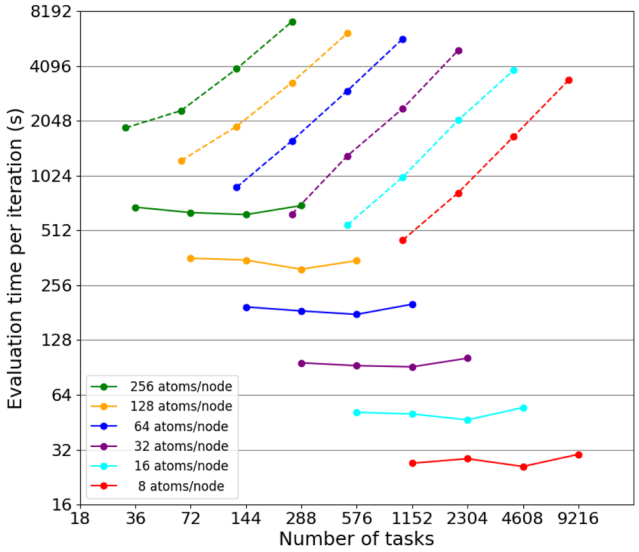

FIG. 6. Weak scaling for the improved (solid lines) and 2015 (dashed lines) implementation of the HSE06 Fock exchange evaluation timings per SCF iteration for GaAs supercells with 256, 512, and 1024 atoms using the *intermediate* FHI-aims species defaults, i.e., 34 NAO basis functions per atom. The gray lines indicate ideal weak scaling. The calculations were run on nodes with 256 GB RAM.

with higher node counts, indicating a sub-optimal parallelization scheme. For the new implementation, this problem has been solved due to the introduction of the additional parallelization layers. As shown in Fig. 6, the run time is close to constant for a given workload.

C. Benchmark calculations for selected systems

Using the algorithm described above, FHI-aims can perform hybrid density functional calculations for both periodic and non-periodic systems. To test the limits of the implementation and document the current runtimes as a reference for future developments, we have selected different systems and geometries. In total, we addressed 18 systems (ten bulk materials, two surfaces, two nanosystems, two clusters, and two molecules). The number of atoms in the systems range from two to more than 30 000 atoms. All the details and references to the original publications discussing these systems and full access to the simulation results are given in the supplementary material. In this work, we only focus on systems that exhibit a gap between the highest occupied and lowest unoccupied states. Testing the limits of the implementation with respect to the number of atoms has helped identify and then resolve issues with order-N^2_{atoms} arrays across different code parts in FHI-aims, which at some point would otherwise dominate the memory consumption. In addition to Tables II and III and the supplementary material, several key results are shown in Fig. 1.

Our test set covers both high- and low-density compounds to explore the limits of the parallelization. In this context, it is important to keep in mind that the local atom density, i.e., the number of atoms within the overlap radius of the basis functions, defines the workload. Formally, the workload scales with $O(n^4_{\text{basis}})$, where n_{basis} is the number of basis functions with non-vanishing overlap. For instance, the simulated carbon allotropes are very dense, so the workload is comparatively high, although each atom has only a few basis functions. In contrast, the ice XI or paracetamol crystals have a low atom density and, in addition, also only few basis functions, so the workload is low. In addition, we also included systems that contain both heavy and light elements, e.g., the inorganic/organic hybrid lead iodide perovskite methylammonium lead iodide (MAPI) or the layered hybrid perovskite phenethylammonium lead iodide (PEPI). These types of systems represent an important computational challenge, i.e., the workload per atom is very inhomogeneous since atoms from very light (H) to very heavy (Pb) elements are included. The refined parallelization scheme over basis functions and the establishment of runtime auto-tuning mechanisms that adapt the workload while running the calculations are critical for efficiently executing these calculations. Finally, we also included magnetic systems, i.e., the Fayalite and Hematite unit cells. Using collinear spin "doubles" the memory and runtime since the spin index in Eq. (4) runs up to two.

FHI-aims provides pre-defined defaults for individual chemical elements that include the integration grids, basis functions and their spatial extent, as well as the expansion order of the mean-field electrostatic potential. As mentioned before, the construction of the auxiliary basis is only based on this default. In this work, we mostly use "light," "intermediate," and "tight" species defaults. Here, we use the defaults denoted as "2020 defaults" in FHI-aims. As the name indicates, they represent defaults with increasing accuracy and also higher computational costs. For a discussion of the accuracy of the "light" and "tight" species defaults, we refer to Ref. 39. We note that the "intermediate" species defaults are the recommended production settings for hybrid DFT calculations in FHI-aims, such as, high quality geometry relaxations, sophisticated band structures, and energy differences. In turn, the calculations presented in Tables II–V with intermediate species defaults can be considered as converged and indicative of realistic simulation settings for DFT simulations of the considered materials. For periodic systems, we also used k-grid densities that can be considered fully converged for systems that provide a gap. We use a k-grid density of $n_i \cdot a_i > 50$ Å, where a_i is the lattice vector length and n_i is the number of k-points along the corresponding k-space direction i. The k-point grids are deliberately chosen to be very well converged, illustrating that the current implementation can be used without *a priori* precision trade-offs in this respect. For example, even for the 2000 atom diamond carbon supercell, a $2 \times 2 \times 2$ k-space grid is used in our benchmark. We emphasize that for most systems even the less computationally demanding "light" settings would lead to qualitatively correct and quantitatively acceptable results, e.g. for geometry optimization or bandgap computations. Similarly, somewhat reduced k-grids would lead to notably lower computational cost and results that can be deemed acceptable. We further note that all the calculations shown here are using runtime parameters, e.g., the number of instances and the Fock matrix block size, which are automatically chosen by the auto-tuning mechanism. As discussed in Sec. III B, this technique aims at yielding very a good computational performance by default and to concurrently avoid out-of-memory scenarios since conservative memory estimates are used. Accordingly, a manual fine tuning of the parameters for the number of instances and the Fock matrix

TABLE II. All the calculations are carried out with the HSE06 ($\alpha = 0, \beta = 0.25$, $\omega = 0.11$ bohr^{-1}) functional. Species: predefined species defaults in FHI-aims (defaults2020). Nodes: number of nodes with 72 cores per node and 256 GB memory. Inst.: number of created instances of the auto-tuning mechanism [cf. Eq. (13)]. Init: initialization time for the Fock exchange computation (only once per SCF cycle). Fock: average HSE06 Fock exchange computation time per SCF iteration. KS: average time per SCF iteration for the solution of the Kohn–Sham equations. M: estimated maximum memory per node in GB. The references point to the publications from which the structural models were taken. The visualization of the structures is shown in the supplementary material.

Bulk systems								
Species	Nodes	Inst.	Init (s)	Fock (s)	KS (s)	No. of basis	No. of states	M (GB)
Boron nitride: BN (2 atoms), 19 × 19 × 19 k-grid, Ref. 52								
Light	1	8	71.384	2.809	0.054	28	12	86.31
Intermediate	1	8	115.769	20.084	0.128	60	12	131.34
Tight	1	3	147.644	84.480	0.389	78	12	158.19
Carbon diamond: C_{2000} (2000 atoms), 2 × 2 × 2 k-grid, Ref. 53								
Light	16	1	45.677	130.690	60.236	28 000	12 000	142.85
	32	4	42.704	53.065	34.734			141.69
Intermediate	64	1	139.876	459.099	57.022	60 000	12 000	207.80
Cubic MAPI: $C_{64}N_{64}H_{384}Pb_{64}I_{192}$ (768 atoms), 2 × 2 × 2 k-grid, Ref. 54								
Light	8	2	68.575	69.039	27.557	15 744	10 912	97.92
Intermediate	8	1	123.644	179.267	32.493	20 672	10 912	127.11
	16	2	83.606	92.080	18.611			98.87
Hematite: Fe_4O_6 (10 atoms), 10 × 10 × 10 k-grid, collinear spin, Ref. 55								
Light	1	8	17.090	26.263	0.263	208	116	93.99
Intermediate	1	3	48.562	83.854	0.410	340	116	93.99
Fayalite: $Fe_8Si_4O_{16}$ (28 atoms), 4 × 7 × 8 k-grid, collinear spin, Ref. 56								
Light	2	8	11.738	18.915	0.923	572	300	85.60
Intermediate	2	2	36.663	92.804	1.544	936	300	93.18
Hydrogen interstitial in silicon: $Si_{64}H_1$ (65 atoms), 4 × 7 × 4 k-grid, Ref. 57								
Light	1	2	28.297	62.990	2.561	1 605	643	117.99
Intermediate	1	1	80.690	225.392	2.588	2 187	643	179.04
	2	2	44.763	112.293	1.337			127.11
Water (liquid): $H_{128}O_{64}$ (192 atoms), 4 × 4 × 4 k-grid, Ref. 57								
Light	1	6	16.157	8.752	2.475	1 536	704	115.58
Intermediate	1	1	102.727	101.951	4.975	3 328	704	193.93
PEPI with defect: $C_{1152}H_{1728}Bi_2I_{288}N_{144}Pb_{69}$ (3383 atoms), 1 × 1 × 1 k-grid, Ref. 1								
Intermediate	16	1	174.454	299.968	69.529	75 100	24 076	209.03
	32	1	140.919	227.803	56.663			131.63
4 × 4 × 4 paracetamol: $C_{4096}H_{4608}N_{512}O_{1024}$ (10 240 atoms), 1 × 1× 1 k-grid, Ref. 58								
Light	16	1	216.257	323.327	380.769	101 888	44 288	214.16
	32	4	207.386	106.763	218.647			205.84
Intermediate	84	1	409.863	803.045	278.179	219 648	44 288	215.74

TABLE II. (*Continued.*)

Species	Nodes	Inst.	Init (s)	Fock (s)	KS (s)	No. of basis	No. of states	M (GB)
Bulk systems								
*Supercell of ice XI: $H_{20384}O_{10192}$ (30 576 atoms), 1 × 1 × 1 k-grid, Ref. 59								
Light	60	5	1418.280	710.858	1670.647	244 608	112 112	443.27
Surfaces								
Graphene on SiC slab: $C_{1108}Si_{432}H_{108}$ (1648 atoms), 2 × 2 × 1 k-grid, Ref. 60								
Light	8	1	76.334	181.153	59.822	26 852	11 184	137.91
Intermediate	32	1	107.521	516.838	32.500	49 116	11 184	141.12
TiO_2 slab: $Ti_{1152}O_{2304}$ (3456 atoms), 1 × 1 × 1 k-grid								
Light	32	2	86.544	149.935	84.533	67 968	35 136	107.82
Intermediate	48	1	176.155	615.024	113.114	115 200	35 136	175.15

block size can further speed up calculations by roughly another 25% for most systems.

1. Bulk systems

The bulk system class is computationally the most challenging one. Due to the 3D periodicity, the neighboring unit cells in the Born–von Karman cell lead to contributions for the Fock-type exchange evaluation in all directions. Boron nitride and the carbon diamond supercell are the densest materials among the bulk test suite with about 0.17 and 0.18 atoms/Å^3, respectively. As discussed earlier, despite a low number of basis functions per atom, this leads to a potentially high workload per basis function since the high atom density increases the number of overlapping basis functions from the neighboring atoms. As a result, the computational effort scales as $O(n_{\text{basis}}^4)$ with n_{basis} the number of nonzero basis functions in a given volume element, since no sparsity can be exploited for basis functions that are placed on top of one another. In Table II, we compare "light" and "intermediate" species defaults for most of the systems in the test suite, where intermediate settings are designed to provide sufficient numerical precision for any production DFT simulations. For boron nitride, we also include the "tight" species defaults, illustrating the scaling with the number of basis functions. The Hematite and Fayalite systems require collinear spin polarization to be included in the calculations in order to obtain the correct electronic structure for their ground states. Both systems have an antiferromagnetic spin ordering. The hydrogen interstitial introduces a shallow defect level that is partially occupied. Among the largest structural models, we include supercells of cubic methylammonium lead iodide perovskite (MAPI) with 768 atoms, a defect complex (a lead vacancy with two Bi substitutions at the Pb site) in the phenylethylammonium lead iodide perovskite with 3383 atoms,[1] and a 4 × 4 × 4 paracetamol supercell containing over 10 000 atoms. The largest system is a supercell of ice XI with over 30 000 atoms (Fig. 1), treated with light species defaults, for which we used nodes with 512 GB memory. For some systems, we indicate the strong scaling behavior by doubling the number of nodes. We observe that the strong scaling for the EXX-contribution evaluation remains intact even for systems beyond 10 000 atoms. For systems for which the number of instances is greater than the number of nodes, more than one instance per node is used. To check the integrity of the algorithm for the largest simulation cell, we compared the binding energy of the Ice XI for the supercell to 30 576 atoms with the result based on the smaller unit cell as described in Sec. II, using a 11 × 11 × 6 k-grid for the light species defaults. We use the HSE06 functional with the vdw-TS dispersion correction.[47] We find a total energy difference of 1.4×10^{-7} eV for the binding energy, which can be considered in perfect agreement within the bounds of the default SCF convergence criteria (see the supplementary material for more details).

2. Surfaces

The simulation of surface slabs is a special case of the 3D periodic systems, in which a vacuum region is inserted in one of the three spatial directions. We selected a hydrogen-passivated $6\sqrt{3} \times 6\sqrt{3}$ silicon carbide slab with nine atomic layers and a graphene sheet at the top (1648 atoms). For this example, the Fermi level is pinned close to the Dirac cone of the graphene sheet, but due to interaction with the slab, the Dirac cone states are partially occupied. In addition, we choose a (100)-oriented TiO_2 slab (3456 atoms) that possesses no metallic surface states. Both systems are large and are demonstrated in simulations using significant resources (32 and 48 nodes, respectively, using intermediate settings). Nevertheless, resources of this kind are available in many high-performance computing centers across the globe and the system sizes shown enable rather realistic simulations of nanoscale processes across chemistry and materials science, without undue interactions across periodic supercell images.

3. Nanosystems

Two different carbon allotropes are chosen: a carbon nanowire and a carbon nanotube. Both systems are one-dimensional in the sense that vacuum has been added in two spatial directions and only one direction is actually periodic. The less bulk-like the simulated system, the lower is the actual workload. This can be directly observed for both nanosystems having the same number of atoms,

TABLE III. All the calculations are carried out with the HSE06 ($\alpha = 0$, $\beta = 0.25$, $\omega = 0.11$ bohr^{-1}) functional. Species: predefined species defaults in FHI-aims (defaults2020). Nodes: number of nodes with 72 cores per node and 256 GB memory. Inst.: number of created instances of the auto-tuning mechanism [cf. Eq. (13)]. Init: initialization time for the Fock exchange computation (only once per SCF cycle). Fock: average HSE06 Fock exchange computation time per SCF iteration. KS: average time per SCF iteration for the solution of the Kohn–Sham equations. M: estimated maximum memory per node in GB. The references point to the publications from which the structural models were taken. The visualization of the structures is shown in the supplementary material.

Nanosystems								
Species	Nodes	Inst.	Init (s)	Fock (s)	KS (s)	n_basis	n_states	M (GB)
Carbon nanotube: C_{2000} (2000 atoms), 1 × 1 × 1 k-grid, Ref. 53								
Light	4	4	41.446	21.119	30.698	28 000	12 000	124.36
	8	8	42.297	11.019	17.887			123.59
Intermediate	8	1	122.634	194.599	42.215	60 000	12 000	168.70
Carbon nanowire: C_{2000} (2000 atoms), 1 × 1 × 1 k-grid, Ref. 53								
Light	8	4	62.211	57.638	20.003	28 000	12 000	112.40
Intermediate	16	1	154.032	549.161	29.219	60 000	12 000	165.05
Clusters and molecules								
Species	Nodes	Inst.	Init (s)	Fock (s)	KS (s)	n_basis	n_states	M (GB)
Solvated DNA: $C_{209}H_{10166}N_{88}Na_{20}O_{5110}P_{20}$ (15 613 atoms), Ref. 61								
Light	32	2	132.269	254.326	546.280	127 328	58 308	180.00
	64	8	129.111	104.250	272.003			188.33
Intermediate	128	1	359.336	736.282	320.969	275 236	58 308	210.79
Water drop: $H_{1200}O_{600}$ (1800 atoms), Ref. 61								
Light	2	4	43.654	14.111	9.484	14 400	6600	107.65
Intermediate	4	1	89.275	84.418	11.444	31 200	6600	131.53
Silicon wire: $H_{246}Si_{460}$ (706 atoms), Ref. 61								
Light	4	4	33.700	42.645	3.805	12 730	5092	90.82
Intermediate	4	2	80.333	134.623	4.814	18 346	5092	126.88
Ac-Lys-Ala19-H^+: $C_{65}H_{112}N_{21}O_{22}$ (220 atoms), Ref. 62								
Light	2	16	6.567	0.862	0.123	2072	904	35.25
Intermediate	2	8	17.415	10.134	0.215	4472	904	82.98

but a very different EXX evaluation time. Comparing for the same number of cores and same species defaults, the nanotube can be evaluated more than five times faster than the nanowire, since the nanotube can be considered a rolled 2D graphene sheet with no bulk-like volume. In contrast, the nanowire has a bulk-like core. Nonetheless, the 2000 atom carbon diamond bulk needs a five times longer evaluation time on the same number of nodes. For either of the nanosystems, quite modest resources are required in view of the system size, i.e., eight and sixteen nodes, respectively.

4. Clusters and molecules

These systems do not have periodic boundary conditions but, as initially mentioned, can still be simulated with the same code as described above. We have selected a part of a DNA molecule solvated in saline water containing overall 15 613 atoms, a water cluster shaped as a sphere ("water drop") containing 1800 atoms, a silicon wire containing 706 atoms and a charged Ac-Lys-Ala19-H+ molecule containing 220 atoms. The DNA system is a remarkably large system for hybrid DFA-based simulations at the numerical precision level of intermediate settings. While 128 nodes for the 15 613 atom solvated DNA molecule are large resources, being able to capture processes in such a system with the accuracy of a hybrid DFA at all is, again, a considerable success. The strong scaling behavior is identical for a metallic system, as can be seen, for example, for the solvated DNA presented in Table III that has fractional occupations because the strong scaling is only dependent on how work is distributed among the different MPI tasks.

In general, we observe that the per-SCF-step evaluation of the EXX matrix shows close-to-ideal strong scaling across the systems. However, the EXX initialization timings do not always scale with the number of nodes. In practice, this has little influence on overall

TABLE IV. Runtime comparison for the HSE06 ($\alpha = 0$, $\beta = 0.25$, $\omega = 0.11$ bohr^{-1}) force (f) and stress (s) evaluation using intermediate species defaults. Mode: Performance of a SCF cycle without force and stress evaluation (. . .), with only force evaluation (f), with force and stress evaluation (f/s). Inst.: number of created instances of the auto-tuning mechanism [cf. Eq. (13)]. Init: initialization time for the Fock exchange computation (only once per SCF cycle). Fock: average HSE06 Fock exchange computation time per SCF iteration. KS: average time per SCF iteration for the solution of the Kohn–Sham equations. Iter: average time for a full SCF iteration. SCF: number of SCF iterations until electronic convergence has been achieved. M: estimated maximum memory per node in GB.

Bulk systems									
Mode	No. of N	Inst.	Init (s)	Fock (s)	KS (s)	Iter (s)	f/s (s)	No. of SCF	M (GB)
Hematite: Fe_4O_6 (10 atoms), 10 × 10 × 10 k-grid, intermediate, collinear spin									
· · ·	1	3	48.562	84.220	0.410	89.041	· · ·	17	89.34
f	1	2	268.248	78.577	0.416	82.642	108.220	18	181.67
f/s	1	2	267.335	75.602	0.410	80.278	575.318	23	200.11
Water (liquid): $H_{128}O_{64}$ (192 atoms), 4 × 4 × 4 k-grid									
· · ·	4	8	39.000	17.913	2.339	24.349	· · ·	11	172.76
f	4	2	65.442	28.589	2.645	35.610	61.176	12	184.58
f/s	4	2	67.330	28.217	2.655	35.783	278.821	15	229.16
Molecules									
Ac-Lys-Ala19-H^+: $C_{65}H_{112}N_{21}O_{22}$ (220 atoms), intermediate									
· · ·	2	8	17.415	10.134	0.214	15.619	· · ·	12	82.21
f	2	2	27.176	11.022	0.228	15.145	52.991	13	87.16

runtimes since the initialization is only evaluated once per SCF cycle and usually does not exceed the cost of an extra SCF step. We provide the number of SCF steps for SCF convergence and the total runtime for each of the calculations in the supplementary material.

D. Computation of forces and stress

We showcase the runtimes for the evaluation of the forces and stress for a small subset of the above-mentioned benchmark systems, as presented in Table IV. As mentioned above, the memory consumption for force and stress computations are considerably larger. We use the same number of nodes for energy, force (f), and force+stress (f/s) evaluation to allow for an unbiased comparison of runtimes. For Fe_2O_3, we find that the computation of the force and stress components increases the memory consumption significantly. However, the difference between a forces-only and a force+stress computation is small. This is due to the fact that the stress components are not computed concurrently but in serial batches; see Sec. III D, which, in turn, results in higher runtimes. The number of SCF iterations also increases, since an additional SCF step is needed for the force evaluation and two additional steps are needed for the stress evaluation. It should be noted that the stress evaluation takes a few additional SCF steps, since FHI-aims slightly tightens the electronic convergence criteria for the stress evaluation by default. Similar behavior can be observed for the water system. It appears that energy and force computations have similar memory consumption, but the additional memory usage of the energy-only computation is due to using more instances. Again, to avoid running out of memory, stress components are computed sequentially.

E. Performance comparison of the PBE, PBE0, and HSE06 functionals

This section focuses on the runtime differences between the semi-local functional PBE and the hybrid density functionals HSE06 and PBE0. We select two bulk materials (hematite: 10 atoms and liquid water: 192 atoms) and a molecule (lysine-terminated polyalanine helix, 220 atoms) to quantify the differences explicitly for each functional. We run the same system for the three functionals on the same number of nodes to allow for an easy runtime comparison, even though PBE and HSE06 would actually need far fewer computational resources than PBE0.

The run-time differences can be significant for dense materials with heavier elements and few atoms—especially between the hybrid density functionals PBE0 and HSE06. The timing for the real-space evaluation of the EXX contribution is directly related to the extent of its Coulomb kernel. The PBE0 functional uses a bare Coulomb potential and, thus, its extent is formally infinite; see Sec. I. In practice, the extent is still limited by the sparsity of the density matrix and by the overlap of the basis functions due to the finite extent of the employed atom-centered orbitals. However, the overlap for the relevant basis pairs in the four-center two-electron integrals extends to several layers of nearest neighbors. In contrast, the HSE06 functional uses a screened Coulomb potential leading to only a short-range EXX contribution; see Sec. I. The "standard" screening parameter for HSE06 is 0.2 Å^{-1}, as given in Ref. 18. In turn, the exchange contribution is limited to pairs of closest neighbors in most practical cases. Thus, the sparsity of all the matrices increases and leads to smaller workloads and hence lower runtimes. As presented in Table V, the difference between PBE0 and HSE06 is most pronounced for dense materials—in our case, the Hematite crystal.

TABLE V. Runtime comparison for the PBE, PBE0 ($\alpha = 0.25$, $\beta = 0.0$), and HSE06 ($\alpha = 0$, $\beta = 0.25$, $\omega = 0.11$ bohr^{-1}) functional for the intermediate species defaults. Inst.: number of created instances of the auto-tuning mechanism [cf. Eq. (13)]. Init: initialization time for the Fock exchange computation (only once per SCF cycle). Fock: average HSE06 Fock exchange computation time per SCF iteration. KS: average time per SCF iteration for the solution of the Kohn–Sham equations. Iter: average time for a single SCF iteration. SCF: number of SCF iterations until electronic convergence has been achieved. M: estimated maximum memory per node in GB.

Bulk systems								
Functional	Nodes	Inst.	Init (s)	Fock (s)	KS (s)	Iter (s)	No. of SCF	M (GB)
Hematite: Fe_4O_6 (10 atoms), 10 × 10 × 10 k-grid, intermediate, collinear spin								
PBE	2	⋯	⋯	⋯	0.191	2.495	15	21.27
HSE06	2	6	27.476	40.527	0.227	43.497	17	96.19
PBE0	2	2	102.815	241.563	0.238	243.340	18	220.02
Water (liquid): $H_{128}O_{64}$ (192 atoms), 4 × 4× 4 k-grid								
PBE	5	⋯	⋯	⋯	1.092	5.515	12	21.51
HSE06	5	10	32.628	14.824	1.209	19.360	11	157.55
PBE0	5	1	34.425	40.576	1.201	45.078	11	156.77
Molecules								
Ac-Lys-Ala19-H^+: $C_{65}H_{112}N_{21}O_{22}$ (220 atoms), intermediate								
PBE	2	⋯	⋯	⋯	0.218	5.372	13	21.82
HSE06	2	8	17.415	10.134	0.215	15.619	12	82.21
PBE0	2	8	16.505	12.803	0.228	18.191	12	75.52

HSE06 is a factor of six faster compared to PBE0. In contrast, for a small molecule, the runtime difference between HSE06 and PBE0 is negligible.

The semi-local GGA functional PBE is faster by a factor of 18 compared to HSE06 for the Hematite crystal unit cell. The differences between HSE06 and PBE decrease for large systems and lighter elements, as well as for less dense materials: this can be observed for both the periodic water system and the molecule Ac-Lys-Ala19-H+. As the system size increases, the direct eigensolver ELPA will eventually dominate the runtimes, while, for smaller systems, the HSE06 exchange evaluation will dominate the runtime. For the largest systems presented in Tables II and III, the PBE runtime can be directly estimated. For those calculations, the eigensolver and the HSE06 exchange evaluation are by far the dominant steps in the calculation so the total HSE06 runtime per SCF iteration is approximately the sum of both. In turn, the PBE runtime can be estimated by using only the timings for the solution of the KS equations. For example, the 4 × 4 × 4 paracetamol supercell is roughly only a factor of 1.5 faster when using PBE compared to the HSE06 calculation.

F. Limiting factors for the weak scaling behavior

The dense storage of the Hamiltonian, overlap, and Fock-type matrices, including the copies needed for the Pulay mixing, grow with $O(n_{\text{basis}}^2)$. Dense storage of these matrices is needed since our examples use a dense eigenvalue solver (ELPA). Therefore, these matrices become the memory bottleneck for large-scale simulations. For the exchange matrix evaluation, we observe the $O(N^2)$ behavior for our largest calculations, i.e., for the Ice XI supercells with up to 30 000 atoms, as shown in Fig. 7. It should be noted that it is well known that all the matrices involved become sparse for very large systems with a gap, in particular the density matrix and the closely related exchange matrix. However, this sparsity has not yet been exploited in the current implementation and will be part of future work.

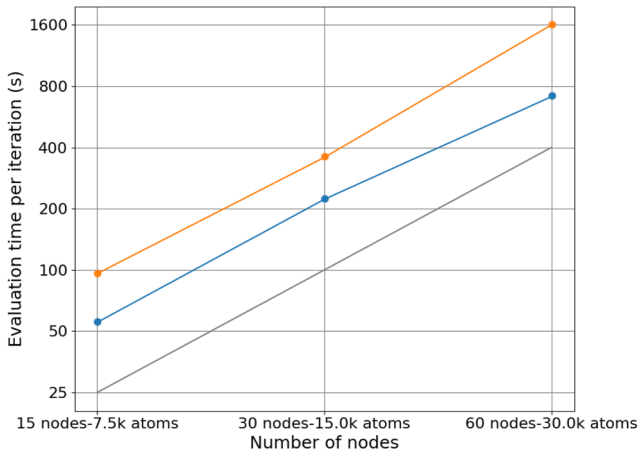

FIG. 7. Weak scaling behavior for the HSE06 Fock exchange evaluation (blue line) and the solution of the KS equations (orange line) for extreme supercells of the Ice XI with 7.5k 15k, and 30k atoms. The graph shows the timings per SCF iteration in seconds in a double logarithmic plot. The horizontal gray lines indicate ideal weak scaling, and the gray diagonal line shows $O(N^2)$ scaling. The calculations were run on nodes with 256 GB RAM (first two data points from left to right) and with 512 GB RAM (last data point), respectively.

V. CONCLUSIONS

For the localized resolution of identity approach to hybrid density functional approximations, RI-LVL,[27,28] as implemented in the FHI-aims code, thorough analysis of memory requirements and code efficiency was performed. An improved distributed storage algorithm was implemented using features of the MPI-3 standard. This allows us to exploit shared memory access on individual nodes for the storage of arrays that are common to each MPI task, while remaining entirely within the MPI paradigm. Other code parts were refactored accordingly, including a more sophisticated load balancing approach that relies on reducing communication by using the available memory to spawn independent "instances" (subgroups of MPI tasks) across which the exchange operator is evaluated. As a consequence, a drastic reduction with respect to memory requirements and a massive increase in the code performance was achieved. For instance, the required memory per node now shows an almost optimal inverse scaling with respect to the number of employed nodes. These improvements are shown to enable the handling of system sizes $\gtrsim$10 000 atoms (more than 30 000 atoms in the largest example considered), extending the reach of hybrid density functional theory on standard CPU-based hardware far beyond the limits of what was previously possible, to our knowledge, in any electronic structure code. Furthermore, the improved distributed storage leads to better load-balance and reduced communication pressure. In turn, this results in an almost perfect, linear scaling of the computational effort with respect to system size and in a virtually ideal speedup with respect to the node count. For large system sizes, these improvements lead to a reduction of the runtimes by two orders of magnitude and more compared to the previous, already optimized implementation in FHI-aims. Regarding the limits of the current implementation, we observe $O(N^2)$ weak scaling behavior for system sizes $\gtrsim$10 000 atoms. Nevertheless, the almost ideal strong scaling behavior is still present even in this regime of extremely large system sizes. The methods and algorithms presented are general and could be implemented in any electronic structure code that relies on localized basis functions.

SUPPLEMENTARY MATERIAL

The file supplementary material contains detailed structural information and references, specifying the sources of the structures. It includes tables with links to data in the NOMAD repository, making all input and output files publicly accessible. In addition, the file provides tables detailing simulation runtimes from start to finish and the number of SCF iterations performed.

ACKNOWLEDGMENTS

This work was funded by the NOMAD Center of Excellence (European Union's Horizon 2020 research and innovation program, Grant Agreement No. 951786) and by the ERC Advanced Grant TEC1p (European Research Council, Grant Agreement No. 740233).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Sebastian Kokott: Investigation (equal); Software (lead); Writing – original draft (lead); Writing – review & editing (equal). **Florian Merz**: Investigation (equal); Software (lead); Writing – original draft (supporting); Writing – review & editing (equal). **Yi Yao**: Software (supporting). **Christian Carbogno**: Investigation (supporting); Software (supporting); Writing – original draft (supporting); Writing – review & editing (equal). **Mariana Rossi**: Investigation (supporting); Writing – original draft (supporting); Writing – review & editing (equal). **Ville Havu**: Investigation (supporting); Writing – original draft (supporting); Writing – review & editing (equal). **Markus Rampp**: Investigation (supporting); Resources (equal); Software (supporting); Writing – original draft (supporting); Writing – review & editing (equal). **Matthias Scheffler**: Funding acquisition (lead); Investigation (supporting); Project administration (equal); Resources (equal); Supervision (equal); Writing – review & editing (supporting). **Volker Blum**: Investigation (equal); Project administration (equal); Resources (supporting); Software (supporting); Supervision (equal); Writing – original draft (supporting); Writing – review & editing (equal).

DATA AVAILABILITY

The FHI-aims code is an academic community code and available to any academic group, including its source code, for a voluntary license fee, enabling, access to the full sources and development thereof by any academic research group. All data that supports this work is openly available in the NOMAD data base at https://dx.doi.org/10.17172/NOMAD/2024.03.13-9. The corresponding URLs for individual data points are listed in the supplementary material.

REFERENCES

[1] H. Lu, G. Koknat, Y. Yao, J. Hao, X. Qin, C. Xiao, R. Song, F. Merz, M. Rampp, S. Kokott, C. Carbogno, T. Li, G. Teeter, M. Scheffler, J. J. Berry, D. B. Mitzi, J. L. Blackburn, V. Blum, and M. C. Beard, PRX Energy **2**, 023010 (2023).
[2] V. W.-z. Yu, J. Moussa, P. Kůs, A. Marek, P. Messmer, M. Yoon, H. Lederer, and V. Blum, Comput. Phys. Commun. **262**, 107808 (2021).
[3] P. Kůs, A. Marek, S. Köcher, H.-H. Kowalski, C. Carbogno, C. Scheurer, K. Reuter, M. Scheffler, and H. Lederer, Parallel Comput. **85**, 167 (2019).
[4] A. Marek, V. Blum, R. Johanni, V. Havu, B. Lang, T. Auckenthaler, A. Heinecke, H.-J. Bungartz, and H. Lederer, J. Phys.: Condens. Matter **26**, 213201 (2014).
[5] R. Schade, T. Kenter, H. Elgabarty, M. Lass, T. D. Kühne, and C. Plessl, Int. J. High Perform. Comput. Appl. **37**, 530 (2023).
[6] A. Nakata, J. S. Baker, S. Y. Mujahed, J. T. Poulton, S. Arapan, J. Lin, Z. Raza, S. Yadav, L. Truflandier, T. Miyazaki *et al.*, J. Chem. Phys. **152**, 164112 (2020).
[7] Z. Luo, X. Qin, L. Wan, W. Hu, and J. Yang, Front. Chem. **8**, 589910 (2020).
[8] J. C. A. Prentice, J. Aarons, J. C. Womack, A. E. A. Allen, L. Andrinopoulos, L. Anton, R. A. Bell, A. Bhandari, G. A. Bramley, R. J. Charlton, R. J. Clements, D. J. Cole, G. Constantinescu, F. Corsetti, S. M.-M. Dubois, K. K. B. Duff, J. M. Escartín, A. Greco, Q. Hill, L. P. Lee, E. Linscott, D. D. O'Regan, M. J. S. Phipps,

L. E. Ratcliff, A. R. Serrano, E. W. Tait, G. Teobaldi, V. Vitale, N. Yeung, T. J. Zuehlsdorff, J. Dziedzic, P. D. Haynes, N. D. M. Hine, A. A. Mostofi, M. C. Payne, and C.-K. Skylaris, J. Chem. Phys. **152**, 174111 (2020).
[9]L. Goerigk and S. Grimme, J. Chem. Theory Comput. **6**, 107 (2010).
[10]F. Tran, G. Baudesson, J. Carrete, G. K. H. Madsen, P. Blaha, K. Schwarz, and D. J. Singh, Phys. Rev. B **102**, 024407 (2020).
[11]A. D. Becke, J. Chem. Phys. **98**, 5648 (1993).
[12]M. Ernzerhof and G. E. Scuseria, J. Chem. Phys. **110**, 5029 (1999).
[13]J. Heyd, G. E. Scuseria, and M. Ernzerhof, J. Chem. Phys. **118**, 8207 (2003).
[14]A. J. Garza and G. E. Scuseria, J. Phys. Chem. Lett. **7**, 4165 (2016).
[15]S. Lany and A. Zunger, Phys. Rev. B **81**, 205209 (2010).
[16]E. Finazzi, C. Di Valentin, G. Pacchioni, and A. Selloni, J. Chem. Phys. **129**, 154113 (2008).
[17]J. Heyd, G. E. Scuseria, and M. Ernzerhof, J. Chem. Phys. **124**, 219906 (2006).
[18]A. V. Krukau, O. A. Vydrov, A. F. Izmaylov, and G. E. Scuseria, J. Chem. Phys. **125**, 224106 (2006).
[19]O. A. Vydrov and G. E. Scuseria, J. Chem. Phys. **125**, 234109 (2006).
[20]R. Peverati and D. G. Truhlar, J. Phys. Chem. Lett. **2**, 2810 (2011).
[21]J.-D. Chai and M. Head-Gordon, J. Chem. Phys. **128**, 084106 (2008).
[22]M. Lorke, P. Deák, and T. Frauenheim, Phys. Rev. B **102**, 235168 (2020).
[23]C. Adamo and V. Barone, J. Chem. Phys. **110**, 6158 (1999).
[24]E. Schwegler, M. Challacombe, and M. Head-Gordon, J. Chem. Phys. **106**, 9708 (1997).
[25]C. Ochsenfeld, C. A. White, and M. Head-Gordon, J. Chem. Phys. **109**, 1663 (1998).
[26]X. Ren, P. Rinke, V. Blum, J. Wieferink, A. Tkatchenko, A. Sanfilippo, K. Reuter, and M. Scheffler, New J. Phys. **14**, 053020 (2012).
[27]A. C. Ihrig, J. Wieferink, I. Y. Zhang, M. Ropo, X. Ren, P. Rinke, M. Scheffler, and V. Blum, New J. Phys. **17**, 093020 (2015).
[28]A. Bussy and J. Hutter, J. Chem. Phys. **160**, 064116 (2024).
[29]A. Förster and L. Visscher, J. Comput. Chem. **41**, 1660 (2020).
[30]W. Hu, L. Lin, and C. Yang, J. Chem. Theory Comput. **13**, 5420 (2017).
[31]J. Dziedzic, Q. Hill, and C.-K. Skylaris, J. Chem. Phys. **139**, 214103 (2013).
[32]X. Wu, A. Selloni, and R. Car, Phys. Rev. B **79**, 085102 (2009).
[33]H.-Y. Ko, J. Jia, B. Santra, X. Wu, R. Car, and R. A. DiStasio, Jr., J. Chem. Theory Comput. **16**, 3757 (2020).
[34]L. Lin, J. Chem. Theory Comput. **12**, 2242 (2016).
[35]H.-Y. Ko, M. F. Calegari Andrade, Z. M. Sparrow, J.-a. Zhang, and R. A. DiStasio, Jr., J. Chem. Theory Comput. **19**, 4182 (2023).
[36]M. Guidon, J. Hutter, and J. VandeVondele, J. Chem. Theory Comput. **6**, 2348 (2010).
[37]L. E. Ratcliff, A. Degomme, J. A. Flores-Livas, S. Goedecker, and L. Genovese, J. Phys.: Condens. Matter **30**, 095901 (2018).
[38]A. Erba, J. Baima, I. Bush, R. Orlando, and R. Dovesi, J. Chem. Theory Comput. **13**, 5019 (2017).
[39]V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter, and M. Scheffler, Comput. Phys. Commun. **180**, 2175 (2009).
[40]X. Ren, F. Merz, H. Jiang, Y. Yao, M. Rampp, H. Lederer, V. Blum, and M. Scheffler, Phys. Rev. Mater. **5**, 013807 (2021).
[41]S. V. Levchenko, X. Ren, J. Wieferink, R. Johanni, P. Rinke, V. Blum, and M. Scheffler, Comput. Phys. Commun. **192**, 60 (2015).
[42]V. Gavini, S. Baroni, V. Blum, D. R. Bowler, A. Buccheri, J. R. Chelikowsky, S. Das, W. Dawson, P. Delugas, M. Dogan *et al.*, Modell. Simul. Mater. Sci. Eng. **31**, 063301 (2023).
[43]M. Benzi, P. Boito, and N. Razouk, SIAM Rev. **55**, 3 (2013).
[44]C. Liu, W. Huhn, K.-Z. Du, A. Vazquez-Mayagoitia, D. Dirkes, W. You, Y. Kanai, D. B. Mitzi, and V. Blum, Phys. Rev. Lett. **121**, 146401 (2018).
[45]J. Y. Park, R. Song, J. Liang, L. Jin, K. Wang, S. Li, E. Shi, Y. Gao, M. Zeller, S. J. Teat *et al.*, Nat. Chem. **15**, 1745 (2023).
[46]F. Knuth, C. Carbogno, V. Atalla, V. Blum, and M. Scheffler, Comput. Phys. Commun. **190**, 33 (2015).
[47]A. Tkatchenko and M. Scheffler, Phys. Rev. Lett. **102**, 073005 (2009).
[48]M. Macher, J. Klimeš, C. Franchini, and G. Kresse, J. Chem. Phys. **140**, 084502 (2014).
[49]P. Pulay, Chem. Phys. Lett. **73**, 393 (1980).
[50]Max Planck Computing and Data Facility, Raven User Guide; accessed 19 June 2023.
[51]J. D. McCalpin, IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter, 19, 1995.
[52]P. P. Villars, L. D. L. D. Calvert, and W. B. W. B. Pearson, *Pearson's Handbook of Crystallographic Data for Intermetallic Phases* (American Society for Metals, Metals Park, OH, 1985).
[53]M. Keçeli, H. Zhang, P. Zapol, D. A. Dixon, and A. F. Wagner, J. Comput. Chem. **37**, 448 (2016).
[54]S. Mohr, L. E. Ratcliff, L. Genovese, D. Caliste, P. Boulanger, S. Goedecker, and T. Deutsch, Phys. Chem. Chem. Phys. **17**, 31360 (2015).
[55]Fe_2O_3 crystal structure: Datasheet from "Pauling file multinaries edition – 2022" in springermaterials, https://materials.springer.com/isp/crystallographic/docs/sd_0314193, copyright 2023 Springer-Verlag, Berlin, Heidelberg, Material Phases Data System (MPDS), Switzerland, and National Institute for Materials Science (NIMS), Japan.
[56]Fe_2SiO_4 ($Fe_2[SiO_4]$) crystal structure: Datasheet from "Pauling file multinaries edition – 2022" in springermaterials, https://materials.springer.com/isp/crystallographic/docs/sd_0375064, copyright 2023 Springer-Verlag, Berlin, Heidelberg, Material Phases Data System (MPDS), Switzerland, and National Institute for Materials Science (NIMS), Japan.
[57]L. Lin, A. García, G. Huhs, and C. Yang, J. Phys.: Condens. Matter **26**, 305503 (2014).
[58]L. H. Thomas, C. Wales, L. Zhao, and C. C. Wilson, Cryst. Growth Des. **11**, 1450 (2011).
[59]A. J. Leadbetter, R. C. Ward, J. W. Clark, P. A. Tucker, T. Matsuo, and H. Suga, J. Chem. Phys. **82**, 424 (1985).
[60]L. Nemec, V. Blum, P. Rinke, and M. Scheffler, Phys. Rev. Lett. **111**, 065502 (2013).
[61]S. Mohr, W. Dawson, M. Wagner, D. Caliste, T. Nakajima, and L. Genovese, J. Chem. Theory Comput. **13**, 4684 (2017).
[62]F. Schubert, M. Rossi, C. Baldauf, K. Pagel, S. Warnke, G. von Helden, F. Filsinger, P. Kupser, G. Meijer, M. Salwiczek *et al.*, Phys. Chem. Chem. Phys. **17**, 7373 (2015).