npj | computational materials

www.nature.com/npjcompumats

Kohn–Sham time-dependent density functional theory with Tamm–Dancoff approximation on massively parallel GPUs

Inkoo Kim[1], Daun Jeong[1], Won-Joon Son[1] ✉, Hyung-Jin Kim[1] ✉, Young Min Rhee[2] ✉, Yongsik Jung[3], Hyeonho Choi[3], Jinkyu Yim[1], Inkook Jang[1] and Dae Sin Kim[1]

We report a high-performance multi graphics processing unit (GPU) implementation of the Kohn–Sham time-dependent density functional theory (TDDFT) within the Tamm–Dancoff approximation. Our algorithm on massively parallel computing systems using multiple parallel models in tandem scales optimally with material size, considerably reducing the computational wall time. A benchmark TDDFT study was performed on a green fluorescent protein complex composed of 4353 atoms with 40,518 atomic orbitals represented by Gaussian-type functions, demonstrating the effect of distant protein residues on the excitation. As the largest molecule attempted to date to the best of our knowledge, the proposed strategy demonstrated reasonably high efficiencies up to 256 GPUs on a custom-built state-of-the-art GPU computing system with Nvidia A100 GPUs. We believe that our GPU-oriented algorithms, which empower first-principles simulation for very large-scale applications, may render deeper understanding of the molecular basis of material behaviors, eventually revealing new possibilities for breakthrough designs on new material systems.

npj Computational Materials (2023)9:81 ; https://doi.org/10.1038/s41524-023-01041-4

INTRODUCTION

Material simulations encounter various problems on different energy, length and time scales. Therefore, a wide range of specific computational methodologies have been designed to yield well-grounded understanding of a wide number of processes occurring in small- and medium-sized organic molecules and large biological systems[1]. At the fundamental level, materials are described by the many-body Hamiltonian in the Schrödinger equation, which is generally considered as an intractable problem owing to the exponential time complexity with size. However, Hohenberg and Kohn[2] has revealed that the electron density possesses a one-to-one correspondence with the ground state wavefunction; based on this notion, the Kohn–Sham formulation of the density functional theory (DFT) drastically simplifies the many-body problem to a set of single-body problems[3]. Supporting a balance between accuracy and efficiency, DFT has been widely utilized in the field of materials science to quantum-mechanically rationalize the molecular factors influencing the structural, electrochemical, and photochemical properties, and its time-dependent extension (TDDFT) has yielded insights into the electronic excited states of diverse materials[4,5].

Over the past decades with the advancement of high-performance computing (HPC) architecture, efficient parallel algorithms have been devised for multi-core central processing units (CPUs), which have facilitated the routine application of DFT methods for large molecules comprising up to a few hundreds of atoms[6–8]. Owing to the intrinsically heavy scaling of DFT to the system size ranging up to the quartic order, systems containing several thousands of atoms have remained virtually unexplored with the conventional DFT methods. Thus, employing cost-reducing approximations such as the linear-scaling methods[9] has been an unavoidable strategy, which inevitably compromises the predictive accuracy to achieve feasibility. With the emerging novel display and battery materials that are characterized by large organic molecules in an amorphous solid-state, researchers require highly efficient simulation platforms that can yield consistent predictive powers for the systems, as in plain DFT.

Recently, material simulations have been facing disruptive changes in HPC architectures driven by the rapidly increasing use of heterogeneous computing accelerators, such as graphics processing units (GPUs)[10,11]. The advent of GPU-programming models has transformed GPUs into general-purpose accelerators, which have solely been considered heterogeneous processors for graphics applications. The compute unified device architecture (CUDA) model[12], for instance, enables direct access to parallel compute units in a GPU with a high level of control by utilizing a grid topology comprising many thread-blocks that are executed concurrently. To fully unlock the potential of new and massively parallel hardware, the pre-existing algorithms should be recast in a form such that parallel execution of kernels with high concurrency is maintained with coordinated data transfers between the host and the accelerators[13].

After the pioneering works on offloading some computationally intensive sections of DFT methods to GPUs[14–16], a series of research attempts have been made to target molecular applications to GPUs, clearly demonstrating the advantageousness of GPU offloading methods. Herein, we will restrict our discussion to the atom-centered Gaussian-type basis sets designed for molecular systems[17], although many studies also reported GPU-based DFT using other basis representations such as plane wave[18] or real-space grids[19] with very different parallelization tactics. Generally, the major computational task in DFT with atom-centered basis can be condensed to the Fock matrix build, which can be categorized into two parts: the contraction of electron-repulsion integrals (ERIs) with a density matrix and the evaluation of the exchange–correlation potential. Following the

[1]Innovation Center, Samsung Electronics, Hwaseong 18448, Republic of Korea. [2]Department of Chemistry, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Republic of Korea. [3]Samsung Advanced Institute of Technology, Samsung Electronics, Suwon 16678, Republic of Korea. ✉email: wonjoon.son@samsung.com; hj.windy.kim@samsung.com; ymrhee@kaist.ac.kr

implementation of the GPU-accelerated Fock build[15,16], several research groups have started developing GPU-specific algorithms for DFT. Because ERI calculations formally exhibit quartic scaling with the size of the system, the concurrent evaluation of the integrals has been a central component of the algorithm development[14,20–27] combined with Fock digestion[28–30]. Efforts were also made to specially prepare the linear-scaling scheme for the exact exchange calculations to work efficiently on GPUs[31,32]. Moreover, the contribution of exchange–correlation potential to the Fock matrix can be efficiently calculated using the grid-batching scheme[33,34]. Recently, ground-state DFT implementations using distributed multi-node multi-GPU systems have been reported[35–37]. However, relatively little attention has been paid to porting TDDFT to multi-GPUs, although most ground-state GPU algorithms can still be applied for the excited-state calculations[38].

Besides, while previous studies have revealed that efficient DFT calculations on GPUs are a promising approach and material simulations are at the stage of up-scaling system sizes, no GPU implementations appear to have achieved a performance on the scale of peta-FLOPS (10^{15} floating-point operations per second) likely with missing sufficiently large implementation that can showcase the capability of heterogeneous hardware. To address the need for large-scale general DFT programs in the field of materials science and to leverage the computing power of massively parallel GPU clusters, we developed a scalable DFT algorithm featuring atomic orbital basis. Moreover, the code was successfully implemented using a multilevel parallel-programming model, targeting hybrid hardware configurations adopting inter- and intra-node connections, as well as accelerators. Our proposed DFT method secured high performance on a state-of-the-art platform over a peta-FLOPS scale with well-behaved load-balancing across the distributed system, while integrating a full stack of simulation capabilities for both electronic ground- and excited-states.

RESULTS

Massively parallel GPU environment

We envisage the integration of distributed Kohn–Sham DFT calculations in high-performance computing systems, which can completely leverage the massive parallelism offered by GPUs. This study employed the CUDA model; hence, throughout this article, we have used CUDA terminology[39]. However, the generalization to other GPU programming models can be readily made (see Supplementary Table 1). Typically, a GPU is based on an array of streaming multiprocessors that are designed to concurrently execute a large number of threads. Threads are fundamental execution units that are processed in parallel on a streaming multiprocessor with a single instruction. Specifically, thread schedulers simultaneously issue each instruction to a warp of threads (typically, 32); accordingly, the same instruction is invariably executed by all the threads in the warp. This feature indicates that different control paths via branching are serialized, thereby adversely affecting the computing performance, referred to as thread divergence. Threads are organized in groups called thread-blocks, which are further organized in a grid structure; therefore, thread and thread-block indices within a grid offer a logical way to invoke computations across the discrete elements in multi-dimensional domains encountered in DFT methods as matrix elements or quadrature points. Moreover, contrary to CPU, a GPU has limited memory capacity and is not expandable. Assuming that a dense matrix is manipulated within the GPU memory, we presume that the practical limit of the matrix dimension lies between 10^4 and 10^5, requiring 0.8–80 gigabytes (GB) of space in random-access memory (RAM) in double precision, constrained by the memory capacity of modern GPUs.

An experimental peta-scale computing system, featuring high-end data-center-grade hardware components, was commissioned using segments of the SSC-21 supercomputer at Samsung Electronics[40]. Figure 1a illustrates the architecture and hardware configuration of the proposed computing system. Each compute node is equipped with dual 32-core AMD EPYC 7543 CPUs (core operating frequency: 2.8 GHz, with 1024 GB RAM), and eight Nvidia A100 GPUs (each with 3456 FP64-cores operating at 1.41 GHz and 80 GB RAM). The GPUs are interfaced with the NVLink connection, supporting bidirectional data-transfer rate of 600 GB/s. The nodes interconnect is arranged in a 5-stage Clos topology, and each node contains four Mellanox InfiniBand HDR fabrics (200 GB/s). Observably, the overhead of data-transfer between the CPU and GPU via PCI-express Gen4 (64 GB/s) represents as an important inefficiency factor that should be mitigated.

In this perspective, as illustrated in Fig. 1b, we propose a multilevel parallel-model based on a massage-passing interface

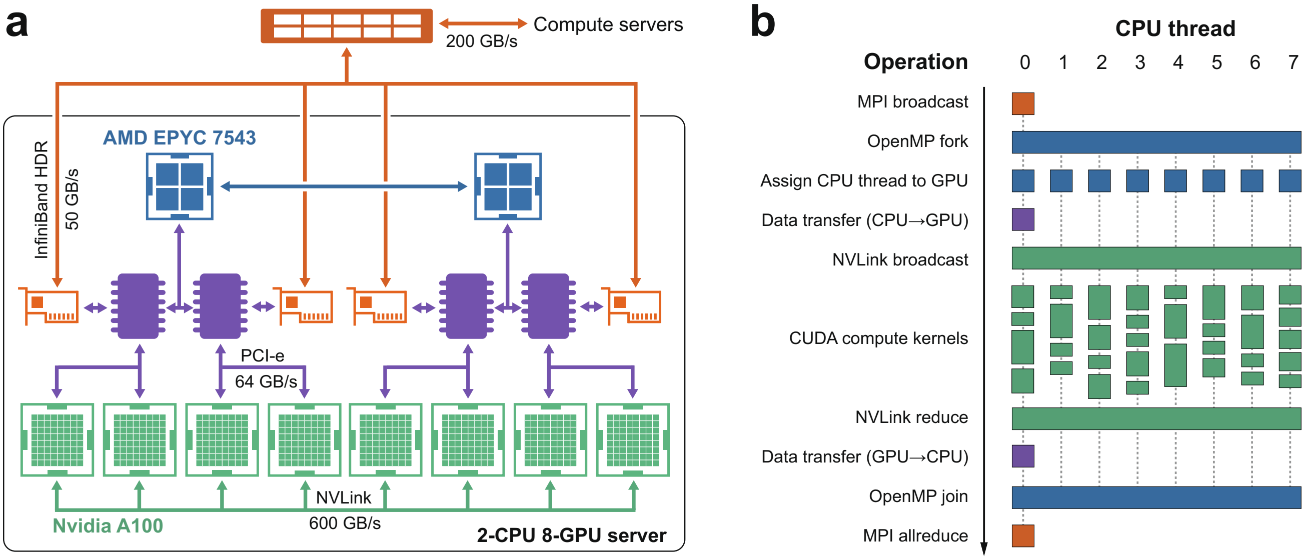


Fig. 1 Massively parallel GPU computing. a Hierarchical structure of compute servers. Theoretical bidirectional data transfer rates within the system are listed. **b** Hybrid MPI-OpenMP-CUDA parallel model for the distributed GPU computing. Each process is color-coded according to the hardware used, as illustrated in **a**.

(MPI) for inter-node communications, open multi-processing (OpenMP) for intra-node CPU/GPU parallelization, and CUDA for GPU offloading of computationally intensive parts of applications. One MPI rank is placed on each node, and by invoking the OpenMP parallel interface, it spawns as many CPU threads as the number of GPUs, each bound to a GPU. To exploit the fast interconnects between intra-node GPUs for data broadcasts and reductions, the input data are initially broadcast via MPI from the master rank to all worker ranks, followed by CPU-to-GPU data transfers solely on the OpenMP master threads, avoiding traffic overload and data congestion in the PCI-express lanes, and the data are instantaneously broadcast to other GPUs via NVLink. The fine-grained GPU computations are grouped within the OpenMP scheduler to ensure a dynamic load-balance. The data-reduction stage proceeds in a manner similar to that for the data-broadcast, but the flow order is reversed: the GPU-private data are first reduced within GPUs in each node using the fast NVLink interconnect, and subsequently, are downloaded to the CPU memory using the OpenMP master threads, followed by data-reduction among the compute nodes via MPI.

As a fundamental requirement for the aforementioned model to operate efficiently and effectively for the DFT methods, the computational task of constructing Fock matrices must not only be translated into fine-grained data-parallel threads, but the optimal data-distribution scheme should also be devised for global load-balancing in multi-GPUs under network configurations. Considering its vitality, this aspect has been presented and discussed in the following sections.

Kohn–Sham equation with a finite basis set

To obtain a comprehensive view of the involved algorithms, a brief summary of the Kohn–Sham DFT and TDDFT methods are presented with an emphasis on the computational aspects. For convenient representation, we have restricted the discussion to closed-shell systems, in which the spin component is integrated from the following equations. Within the Born–Oppenheimer framework, the non-relativistic many-body Hamiltonian in the Schrödinger equation for polyatomic molecules can be transformed into a set of effective one-electron Kohn–Sham equations as follows[3]:

$$\hat{F}\psi_i(\mathbf{r}) = \varepsilon_i\psi_i(\mathbf{r}) \tag{1}$$

where $\hat{F} \equiv \hat{T} + \hat{V}_{\mathrm{KS}}$ denotes the Fock operator comprising the non-interacting kinetic energy operator, $\hat{T}$, and the Kohn–Sham operator, $\hat{V}_{\mathrm{KS}}$. The effective potential given by $\hat{V}_{\mathrm{KS}}$ generates independent Kohn–Sham (KS)-orbitals, ψ_i, of energy ε_i, from which the density can be obtained as $\rho(\mathbf{r}) = 2\sum_i|\psi_i(\mathbf{r})|^2$. Equation (1) is solved via iteration through minimizing the total electronic energy $E = E_T + E_V + E_J + c_X E_K + E_{XC}$, where the respective energy components represent the kinetic, nuclear attraction, Coulomb, Hartree–Fock (HF) exchange, and the exchange–correlation (XC) energies, stemming from the description of $\hat{F}$ (refer to Supplementary Text 1) with respect to a set of KS-orbitals. In particular, hybridizing a portion of the exact HF exchange with the mixing parameter c_X is currently a de facto strategy for improving the accuracy in the density functional approximation[7]. Generally, the approximate forms of the functional E_{XC} are formulated under the generalized gradient approximation (GGA)[41] as $E_{XC} \equiv \int f(\rho_\alpha, \rho_\beta, \nabla\rho_\alpha, \nabla\rho_\beta)\,d\mathbf{r}$, which depends on spin densities and their gradients. With $c_X = 1$ and $E_{XC} = 0$, E is rendered as the general Hartree–Fock energy expression.

Linear combinations of finite non-orthogonal basis sets corresponding to contracted Gaussian functions $\{\phi_\mu\}$ centered on atoms are utilized as in conventional quantum chemistry methods[42], which are hereafter referred to as atomic orbitals (AOs). This strategy describes the KS-orbitals as $\psi_i(\mathbf{r}) = \sum_\mu C_{\mu i}\phi_\mu(\mathbf{r})$. Thus, Eq. (1) can be rewritten in terms of AOs as the following non-linear eigenvalue problem[43–45]:

$$\mathbf{FC} = \mathbf{SC}\varepsilon \tag{2}$$

where $\mathbf{F}$ and $\mathbf{S}$ are the Fock and overlap matrices, respectively, $\mathbf{C} \equiv [\mathbf{C}_o, \mathbf{C}_v]$ denotes the KS-orbital coefficient matrix with N_o-occupied and N_v-virtual orbital columns, and $\boldsymbol{\varepsilon}$ symbolizes the diagonal matrix with ε_i. The density can also be represented in AO basis as the one-electron density matrix, $\mathbf{P} = 2\mathbf{C}_o\mathbf{C}_o^T$. All matrices possess $N \times N$ dimensions, where N indicates the total number of basis functions.

The Fock matrix can be decomposed into three parts[45]: the one-electron core Hamiltonian, $H_{\mu\nu}^{\mathrm{core}}$, comprising the kinetic and nuclear attraction integrals, the two-electron part of the Fock matrix, $G_{\mu\nu}$, including the Coulomb and HF exchange integrals, and the XC contribution to the Fock matrix, $F_{\mu\nu}^{\mathrm{XC}}$. Namely, we have

$$F_{\mu\nu} = H_{\mu\nu}^{\mathrm{core}} + G_{\mu\nu}[\mathbf{P}] + F_{\mu\nu}^{\mathrm{XC}}[\rho(\mathbf{r})] \tag{3}$$

and note that different representations of the density are used here. The explicit expressions for these matrices are listed in Supplementary Text 1. Because the $H_{\mu\nu}^{\mathrm{core}}$—independent of the density—is invariant during the calculations, it is computed once at the beginning of the calculation, whereas the density-dependent $G_{\mu\nu}$ and $F_{\mu\nu}^{\mathrm{XC}}$ are re-evaluated at every iteration. Typically, the Fock matrix is transformed at each iteration into an orthogonal basis for facile diagonalization, and the eigenvectors are subsequently back-transformed to yield the KS-orbitals and density. The updated Fock matrix is formed using the new density in Eq. (3), and this process is repeated until self-consistency of the ground-state density is attained.

The excited states in the DFT formalism can be accessed through the time-dependent evolution of ground-state KS-orbitals[46,47]. Within the adiabatic approximation, in which the explicit time dependence of the XC functional is neglected, and within the linear-response formalism under the Tamm–Dancoff approximation (TDA)[48], the eigenvalue equation for the excitation energies as poles is expressed as[49]

$$\mathbf{AX} = \boldsymbol{\omega}\mathbf{X} \tag{4}$$

where $\mathbf{A}$ represents the Hamiltonian in the space of singly-excited electronic configurations, and $\mathbf{X}$ and $\boldsymbol{\omega}$ represent the excitation amplitudes and energies, respectively. Although the full TDDFT formulation with the deexcitation amplitudes affords slightly more accurate transition dipole moments between the ground and excited states[50], TDA typically retains a good agreement with it both in terms of excitation energies and molecular properties. In addition to the moderate speed-up by neglecting the deexcitations, which renders TDA suitable for large-scale applications, the triplet instability problem in TDDFT is also generally rectified by TDA often with significantly improved triplet excitation energies[51]. With the closed-shell reference ground-state, two types of excited-state solutions can be derived from Eq. (4), corresponding to the spin multiplicity of singlet or triplet, and the explicit expressions are summarized in the Supplementary Text 2.

According to the single-electron excitation from occupied to virtual orbitals, $\mathbf{A}$ possesses a dimension of $N_oN_v \times N_oN_v$, which can become prohibitively large for full diagonalization with increasing number of basis functions. In this context, the Davidson method allows for the extraction of the lowest few excitations of interest for practical applications via constructing a subspace Hamiltonian[52,53]. The subspace Hamiltonian is iteratively diagonalized to produce the lowest eigenvalues while incrementing the subspace $\mathbf{b} \equiv \{\mathbf{b}_1, \mathbf{b}_2, \cdots\}$ within tractable limits. Thus, a computational bottleneck encountered here involves building a matrix-vector product $\boldsymbol{\sigma}_k \equiv \mathbf{A}\mathbf{b}_k$ from which the subspace Hamiltonian matrix element can be conveniently obtained as $H_{kl} = \mathbf{b}_k^T \cdot \boldsymbol{\sigma}_l$. The initial subspace vectors can be constructed using a set of orthogonal vectors of $\boldsymbol{e}_{ia}$ where i and a denote the occupied and virtual orbital indices, respectively. The matrix element σ_{ia} can be expressed in AO basis with the Fock-type matrix $\tilde{F}_{\mu\nu}$ as[54]

$$\sigma_{ia} = (\varepsilon_a - \varepsilon_i)b_{ia} + [\mathbf{C}_\mathrm{O}^\mathrm{T}\tilde{\mathbf{F}}\mathbf{C}_\mathrm{V}]_{ia} \tag{5}$$

$$\tilde{F}_{\mu\nu} = \tilde{G}_{\mu\nu}[\tilde{\mathbf{P}}] + \tilde{F}_{\mu\nu}^{\mathrm{XC}}[\rho(\mathbf{r}), \tilde{\mathbf{P}}] \tag{6}$$

where the scaled occupied–virtual density matrix $\tilde{\mathbf{P}} = \mathbf{C}_\mathrm{o}^\mathrm{T}\mathbf{b}_k\mathbf{C}_\mathrm{v}$, which is generally non-symmetric in nature, was used instead to compute $\tilde{G}_{\mu\nu}$. The similarity between $\tilde{F}_{\mu\nu}$ and $F_{\mu\nu}$ in Eq. (3) allows the former to be calculated with the corresponding ground-state implementation with slight modifications. The integral transformation between atomic and molecular orbital bases can be represented as matrix multiplications after constructing $\tilde{F}_{\mu\nu}$, which can be efficiently performed using a multi-GPU linear-algebra library. Two large vectors $\mathbf{b}_k$ and $\boldsymbol{\sigma}_k$ must be stored in the CPU memory at each Davidson iteration. Modern computers incorporate RAM close to or in the order of terabytes, and storing the matrices in the core memory poses no challenges.

In DFT and TDDFT calculations, a major computational bottleneck arises from the construction of $G_{\mu\nu}$ and $\tilde{G}_{\mu\nu}$, respectively, both involving ERIs over four-center AOs, defined as

$$(\mu\nu|\lambda\sigma) = \iint \frac{\phi_\mu(\mathbf{r})\phi_\nu(\mathbf{r})\phi_\lambda(\mathbf{r}')\phi_\sigma(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\,\mathrm{d}\mathbf{r}\mathrm{d}\mathbf{r}' \tag{7}$$

exhibiting a formal $\mathcal{O}(N^4)$ complexity with respect to the basis set size for a given molecule. In practice, the number of significant ERIs scales asymptotically as $\mathcal{O}(N^2)$ with the molecule size because of the locality of the Gaussian-type functions. Additionally, a secondary bottleneck with a $\mathcal{O}(N^3)$ complexity arises from the calculation of $F_{\mu\nu}^{\mathrm{XC}}$, which requires numerical quadrature in real space. Which of these two bottlenecks is dominant is decided by the size of the system and the basis set of choice, and we will deal with their computational aspects in more details in the following sections. Because matrix operations also frequently occur in the calculations, by employing the high-performance GPU linear algebra library, we performed a cursory survey on the performance of matrix multiplications and diagonalizations up to a size that can completely occupy the GPU memory (refer to Supplementary Table 2). Even with their inherent $\mathcal{O}(N^3)$ complexity, mostly owing to exceptionally low prefactor, their computations can be effectively hidden in the timescale of DFT calculations, which is predominated by the Fock build.

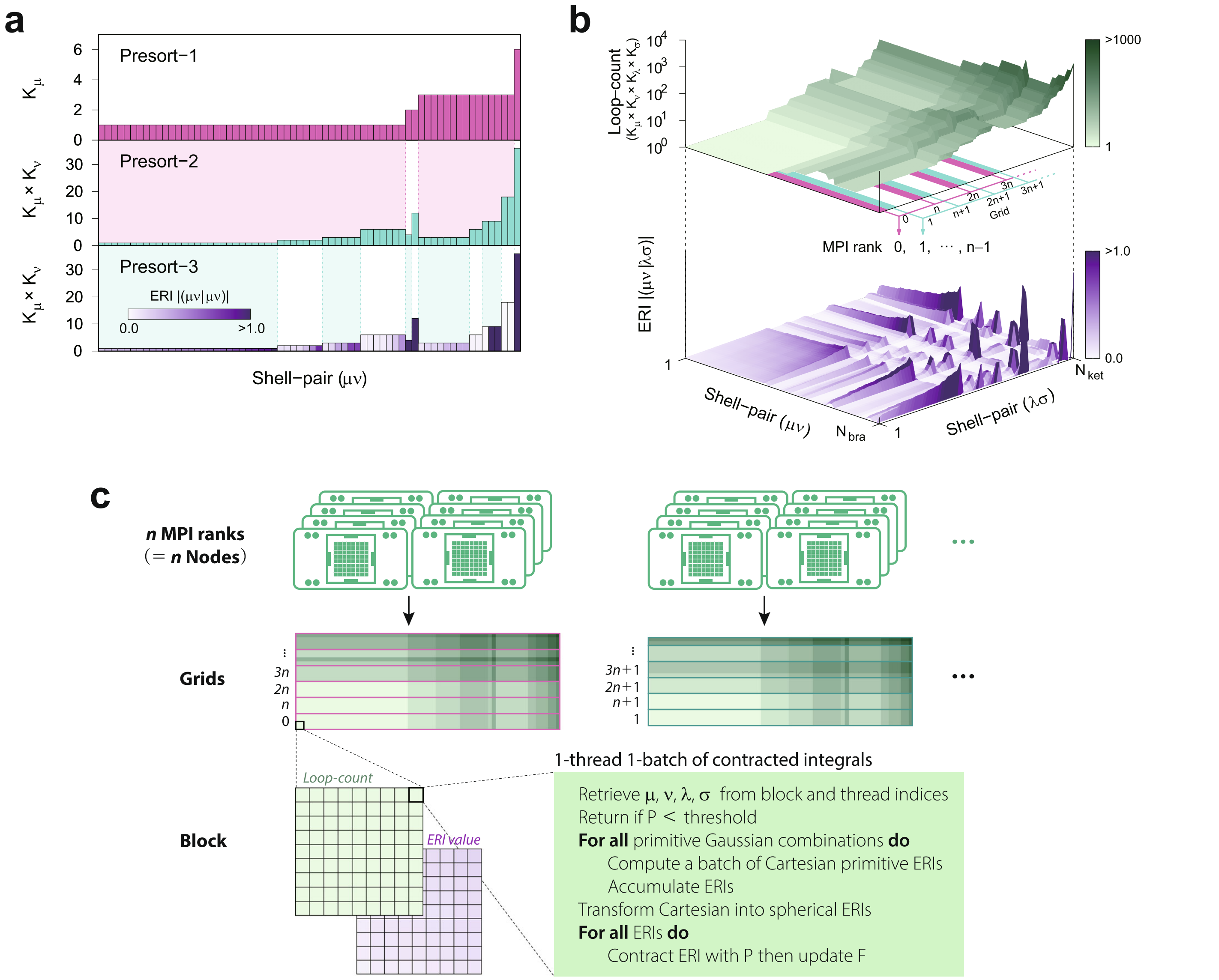
Direct Fock matrix build on multi-GPUs

The time-consuming task of constructing $G_{\mu\nu}$ and $\tilde{G}_{\mu\nu}$ in the respective DFT and TDDFT calculations entails the Coulomb and HF exchange contributions, which are generally computed in an integral-driven and direct scheme[55] (refer to Supplementary Text 3). As explained in the above, the number of significant ERIs scales asymptotically as $\mathcal{O}(N^2)$, and as a result, there exists a crossing-point at which matrix operations with $\mathcal{O}(N^3)$ complexity become computationally dominant. However, the large prefactor in ERI evaluations with higher angular momenta functions and the spatial distribution of the basis functions obfuscate pinpointing such a crossing-point. For achieving an efficient Fock build with multi-GPUs, a concurrent evaluation for the batches of ERIs is a key requirement. Certain ERI algorithms, such as McMurchie and Davidson[56], Head-Gordon and Pople[57], and Rys[58], were applied as GPU extensions[14,15,20,23,30]. However, as widely anticipated, ERI implementations based on recursion relations are generally suited for treating low angular momenta integrals featuring AOs of $l \leq 1$, largely due to the limited memory of GPU, while the Rys quadrature scheme, which computes the ERI by n-point numerical integration, where n is determined by the half-sum of the angular momenta of the AOs, is regarded suitable for GPU implementation, and hence, is adopted in the present work.

We designate a shell as a group of $2l+1$ AOs in spherical harmonics arising from the given basis function, denoted hereafter as $\mu \equiv \{\mu_0, \cdots, \mu_{2l+1}\}$, and therefore, $(\mu\nu|\lambda\sigma)$ represents a batch of $(2l_\mu+1)(2l_\nu+1)(2l_\lambda+1)(2l_\sigma+1)$ ERIs, which is processed by a single GPU thread. The AOs inside the bra $(\cdot|$ or ket $|\cdot)$ of ERIs are interchangeable with each other, and accordingly, for instance with AOs of $l \leq 2$ for brevity, only six groups of distinct shell-pairs exist: (ss), (ps), (pp), (ds), (dp), and (dd). Here, negligible shell-pairs can be eliminated by examining their pre-exponential factors to the ERIs, thus reducing the computation effort significantly[59], in particular, for large molecules where the AO pairs are more prone to negligible overlap conditions. As a sufficiently small value is crucial to prevent accumulation of errors and potential linear dependency in the basis set with large molecules, shell-pairs exhibiting pre-exponentials under 10^{-13} au are discarded in our implementation.

After utilizing a permutational symmetry between the bra and the ket, any symmetry-related ERI sub-block can be defined by combining two groups of these shell-pairs (Supplementary Fig. 1). For best GPU performance, each sub-block should be computed separately as the differing angular momenta combinations of the involved AOs clearly leads to conditional branching, which hampers the concurrent processing of ERIs. Further and more importantly, arranging each shell-pair in advance is a critical measure for ensuring maximum concurrency within a thread-block. Otherwise, because each shell is a spherical set of contracted Gaussian-type functions, each with varying numbers of primitive functions that are viewed as a nested loop with different lengths, can potentially lead to thread divergence. Evidently, all threads within a thread-block should ideally run over identical loop-structures to avoid thread divergence. In accordance with the scheme reported in[33], we rearranged the shell-pairs at three levels as depicted in Fig. 2a. The $(\mu\nu)$ shell-pairs are initially sorted into subgroups according to the outer loop count (i.e., contraction number K_μ). Subsequently, the shell-pairs within each subgroup are sorted again according to K_ν, thereby leading adjacent shell-pairs to have likely the same loop structure and counts. Finally, for each subgroup with the same configuration of K_μ and K_ν, the shell-pairs are further sorted according to the Schwarz upperbound $\sqrt{(\mu\nu|\mu\nu)}$. This ordering, as explained hereafter, allows the threads in a thread-block defined with a judicious size to likely possess the same loop structure with similar ERI upperbounds. The computation of some thread-blocks can also be skipped when the associated densities are smaller than a predefined threshold (e.g., 10^{-10} was used in this work). This rearrangement of the shell-pairs can be performed simply by sorting a suitable function such as $s(\mu,\nu) \equiv 10^6 \cdot K_\mu + 10^3 \cdot K_\nu + \sqrt{(\mu\nu|\mu\nu)}$ where the multiplicative factor clearly indicates the sorting level.

Figure 2b depicts the loop-count and upperbound of the ERI batches in a sub-block after the above sorting process. Clearly, the computational loads represented by the loop-count are mostly concentrated on the upper-right corner of the sub-block (i.e., the region where both shell-pair indices approach their last elements). To achieve an optimal load balance of asymmetrical data across distributed GPUs, we directly mapped the ERI sub-block, which is characterized by $N_\mathrm{bra} \times N_\mathrm{ket}$ shell-pair combinations, onto the grid topology by dividing into $N_\mathrm{g} \equiv N_\mathrm{bra}/N_\mathrm{t} \cdot N_\mathrm{b}$ grids, each with a dimension of $(N_\mathrm{b}, N_\mathrm{ket}/N_\mathrm{t})$ thread-blocks, again each with a dimension of $(N_\mathrm{t}, N_\mathrm{t})$ threads. In other words, $(N_\mathrm{t}, N_\mathrm{t})$ threads form a thread-block, and $(N_\mathrm{b}, N_\mathrm{ket}/N_\mathrm{t})$ thread-blocks form one grid element, which spans over $N_\mathrm{b} \cdot N_\mathrm{t}$ shell-pairs of the bra and all shell-pairs of the ket. Thus, in a GPU kernel, a grid evaluates a total of $N_\mathrm{b} \cdot N_\mathrm{t} \cdot N_\mathrm{ket}$ batches of ERIs over the spherical harmonics manifold, and N_g grids cover the entire ERI sub-block. This strategy of partitioning into rectangular grids ensures a balanced data distribution among the compute nodes in a two-dimensional



```
Retrieve μ, ν, λ, σ from block and thread indices
Return if P < threshold
For all primitive Gaussian combinations do
    Compute a batch of Cartesian primitive ERIs
    Accumulate ERIs
Transform Cartesian into spherical ERIs
For all ERIs do
    Contract ERI with P then update F
```

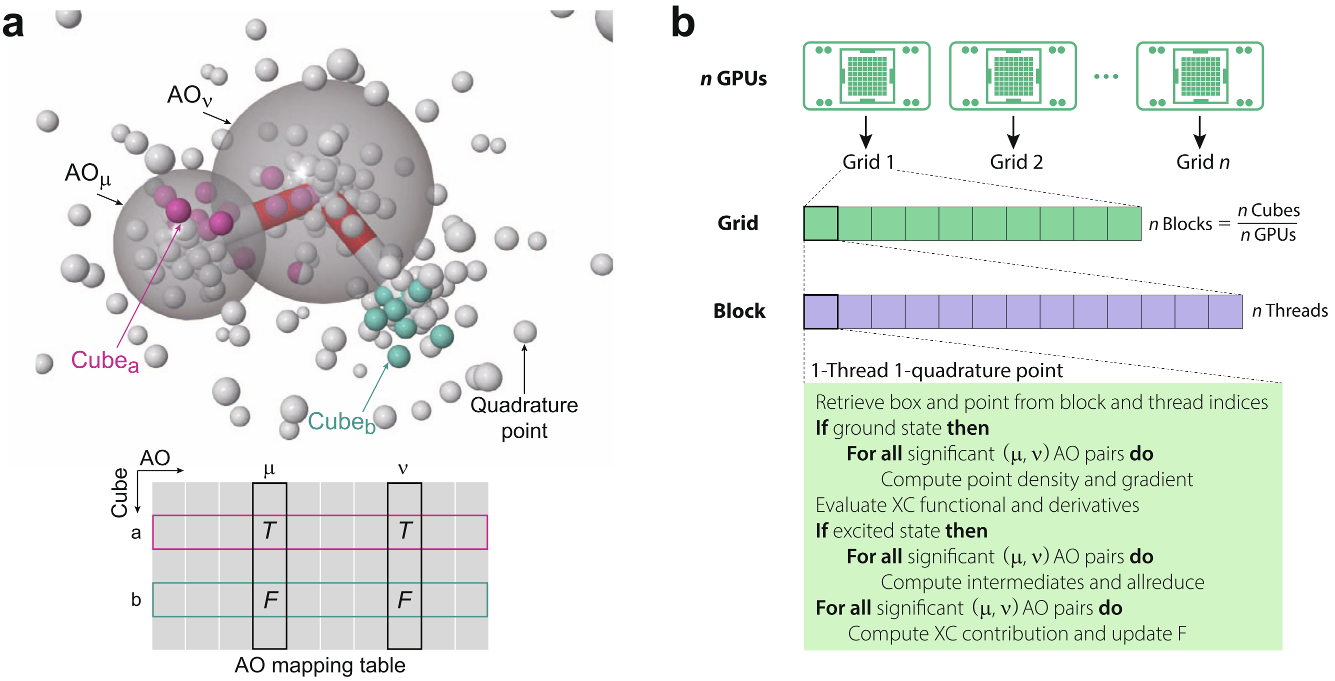
Fig. 2 Multi-GPU mapping of electron repulsion integrals (ERIs) for Fock build. a 3-Level presorting scheme for shell-pairs. The case with the (*ss*) shell-pairs from water with the def2-TZVP basis set is shown. Subgroups subject to subsequent presorting are colored in alternating shades for clarity. **b** The (*ss*|*ss*) sub-block of ERIs using the presorted shell-pairs illustrated in **a**. The vertical scales (loop-counts and estimated ERIs) are visually supplemented by the coloring schemes on the side. Grid partitioning scheme for MPI ranks is also shown. **c** Balanced mapping of grid arrays to multi-GPUs. Grid arrays constructed in **b** are shown. Each block comprises two-dimensional threads with similar loop-counts and ERI values.

block-cyclic layout of the sub-block of ERIs, as will be explained in the following paragraph.

Given $N_g \gg N_r$ with N_r being the number of MPI ranks, the loop over the grids with a stride of N_r can distribute the computational load evenly across MPI ranks; this is aided especially when N_t and N_b are relatively small to allow for data locality such that the N_r grids distributed among the MPI ranks collectively map on regions of similar computational load as a whole (Fig. 2c). Each compute node is assigned with N_g/N_r grids, and the load-balancing among the intra-node GPUs can be addressed dynamically within the OpenMP parallel-model. The OpenMP scheduler ensures full-occupation of the GPUs by continuously offloading the subsequent grid to the first available GPU until all the grids are computed. Moreover, not only the sorting process ensures that the concurrency of the threads is guaranteed within a thread-block as each thread will likely possess the same or at least similar loop structures, but the similar upperbound further permits that the computation of the thread-block is skipped at the beginning of the GPU kernel when the associated density matrix element is negligible, further improving the efficiency.

Numerical multi-center XC-grid integration

In this section, we present the calculations of the XC contributions to the energy and the Fock matrix using numerical quadrature in real space, with the formulations given in Supplementary Text 4. The quadrature points and their relative weights are generated around each nucleus through the combination of radial and angular grids using Becke's partitioning scheme[60], in which we controlled the radial partitioning via the Euler–Maclaurin[61] and the angular parts by using the Lebedev quadrature[62]. Under the GGA formalism, to evaluate E_{XC} and its derivatives at every quadrature point, $\mathbf{r}_q$, for any given iteration, the point electron density, ρ_q, and its gradient, $\nabla \rho_q$, are required, which are generally computed on-the-fly in order to avoid storing the value, ϕ^q_μ, and the gradient, $\nabla\phi^q_\mu$, of all AOs requiring $4N \cdot N_q$ data points wherein the number



Cube	μ	ν
a	T	T
b	F	F

```
1-Thread 1-quadrature point
Retrieve box and point from block and thread indices
If ground state then
    For all significant $(\mu, \nu)$ AO pairs do
        Compute point density and gradient
Evaluate XC functional and derivatives
If excited state then
    For all significant $(\mu, \nu)$ AO pairs do
        Compute intermediates and allreduce
For all significant $(\mu, \nu)$ AO pairs do
    Compute XC contribution and update F
```

Fig. 3 Multi-GPU mapping of numerical quadrature for exchange–correlation. a Schematic illustration of the partitioning scheme of the quadrature points. Data were generated for water with (10, 14)-grid and partitioned into 44 boxes. See Method for the grid definition. Points from two select boxes are shown in magenta/green. Two *s*-type AOs are displayed in gray; the boundary implies their significance threshold. Significance table with a true or false value with the related elements entailing the overlap between boxes and AOs is given at the bottom. **b** Evenly-distributed mapping of boxes to multi-GPUs. Each grid, comprising a one-dimensional array of blocks corresponding to boxes, is scheduled to a single GPU. Each block contains threads corresponding to points with the predetermined list of AO contribution in accordance with the significance table.

of quadrature points, N_q, are generally greater than the number of AOs, N. Thus, for a given $\mathbf{r}_q$, two loops over the AO indices, ϕ_μ and ϕ_ν, must be completed to calculate the pointwise XC contribution to $F_{\mu\nu}$. However, the double-loop clearly leads to thread divergence as the quadrature points that map onto the threads are naively generated by three-loops over the atomic, radial, and angular indices, and accordingly, are practically dispersed in the space, with different contributions that primarily originate from the surrounding AOs.

To ensure concurrency and efficiency in the GPU kernel for the XC-grid integration, the quadrature points are divided into N_c cubes, each collecting the points in the vicinity. Accordingly, the AOs that make non-zero contributions to these quadrature points are rendered almost identical when the volume of the cube is sufficiently small; this condition can be realized by recursively dividing the cube encompassing the entire molecule into octants until the number of points in a single or multiple octants at a given depth is less than a number of threads to be treated in the thread-block. Similar approaches have been discussed in the literature[33,34].

Figure 3a illustrates our proposed approach focusing on the quadrature points over a water molecule as an example. The quadrature points from two selected cubes are displayed; one set of quadrature points is located near one of the H atoms, whereas the other set is distributed along an O–H bond. The hypothetical *s*-type AOs of the O and H atoms are also indicated in the figure. The boundary of the AOs represents the surface beyond which the AO value is treated as negligible (i.e., $\phi_\mu \to 0$). In the case of AO with a higher angular momentum, the largest value among the $2l+1$ spherical components is employed to define such boundaries. If at least one point in a cube is encompassed by such a boundary, we include the corresponding AO in the AO-mapping matrix of the dimension of $N_c \times N$, for which we utilized the smallest 2-byte data-type, as supported by CUDA, requiring $2 \cdot N_c \cdot N$ bytes in total. As two AO loops inside each thread are required in the direct approach to calculated ϕ_q and $\nabla\phi_q$, predetermination of the significant AOs for the given cube is an imperative condition for achieving computational efficiency as all threads in a thread-block undergo the same computations, involving just the surrounding AOs.

Because similar computational workloads cannot generally be expected from each cube, we evenly divided the list of cubes to each GPUs (Fig. 3b). However, even such proportional distribution of the cubes to all GPUs has been proven to exhibit optimal scalability for large-scale calculations as observed in our findings in later sections. Furthermore, in TDDFT, considering that ρ_q and $\nabla\rho_q$ are invariant throughout the TDDFT calculations, they are calculated only once for each quadrature point since the converged ground-state density is used throughout, and subsequently, stored in the core memory for GPU offloading for instant access when needed. We note that the GPU memory is generally sufficiently large to store these values for various grid parameters (Supplementary Table 3).

Relative performance of GPU versus CPU

A fair comparison is difficult to obtain because of the heterogeneity in microarchitectures. However, to enable a reasonable and pragmatic comparison of GPU versus CPU performance on the DFT methods, we examined the DFT and TDDFT timings measured with our GPU code and a well-established CPU code[63] on the commodity GPU and CPU computing servers, respectively. Moreover, we refrain from comparing GPU performance with single-threaded CPU performance since this approach does not represent the practical conditions in materials simulation as multi-threaded executions are more commonplace. To this end, we surmised that for commodity hardware, a GPU server will normally contain two Nvidia A100 devices with a total theoretical peak (R_{peak}) of 19.5 tera-FLOPS, whereas a CPU server with dual 32-core

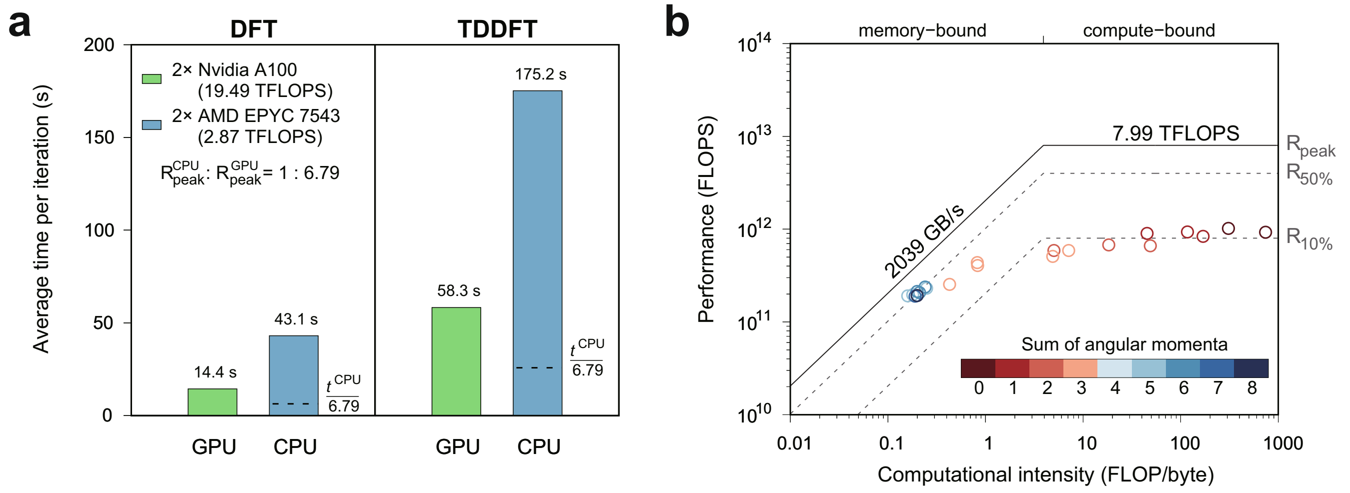


Fig. 4 Performance of 2-GPU versus 2-CPU server. **a** Average wall-time for a single iteration in DFT and TDDFT for 5TCzBN molecule. Horizontal dotted lines indicate the ideal wall-times that would have been obtained when our GPU code exhibited an identical FLOP performance to the CPU code. **b** Roofline analysis of GPU kernel performance corresponding to the Fock build for a single Nvidia A100 GPU. Here, 21 compute kernels depending on the distinct combinations of angular momenta are shown. The theoretical peak performance was bound by the profiling conditions of the employed tool.

AMD EPYC 7543 processors will possess a total R_{peak} of 2.9 teraFLOPS. Based on the ratio of the theoretical peak performances between the GPU and CPU servers, we estimate an ideal speed improvement of 6.8.

As a practical probe-molecule for benchmarks, we selected a blue-emitting organic light-emitting diode (OLED) material, 5TCzBN[64], comprising 233 atoms and 2,182 basis functions under the def2-SVP basis sets (Supplementary Fig. 2). The Becke's three-parameter hybrid exchange functional ($c_X = 0.2$) with the Lee–Yang–Parr correlation functional (B3LYP) was employed[65–67]. Approximately identical energy eigenvalues for both the ground and excited states were obtained for the two independent DFT implementations, thereby validating our implementation (Supplementary Table 4). One may wonder that the discrepancy in the SCF energies in this table (~44 μH) is larger than ideal. This is due to the difference in the grid point definitions between Turbomole and our code. Indeed, additional HF-SCF calculations for 5TCzBN with the same basis set produced a much reduced discrepancy of less than 1 μH.

As demonstrated in Fig. 4a, the obtained speed improvements in the average time per iteration, t_{it}, are 3.2 and 2.6 for DFT and TDDFT, respectively, on the GPU server equipped with two A100 in reference to the CPU server running 64 threads. These speed improvements correspond to 47% and 38% of the ideal value based on the theoretical peak. To further characterize the GPU performance, the roofline analysis using the Nsight profiling toolkit is plotted in Fig. 4b, clearly visualizing the 21 GPU kernels distributed in both memory and compute-bound regions. The performance of kernels involving larger angular momenta are limited by the memory bandwidth due to a larger number of intermediate integrals. In the compute-bound region, approximately 10% of the peak performance is drawn, suggesting a potential for enhancement through further code optimizations. Notably, the lower performance may be partially articulated with the larger prefactor in the Rys quadrature scheme in the ERI evaluations. The t_{it} for TDDFT is approximately four times larger than that of DFT because the three roots, entailing an equal number of the Fock matrix builds, were computed sequentially. Although simultaneously solving for all roots is generally considered more efficient[53], we have resorted to the sequential algorithm to minimize the memory usage since the simultaneous approach necessitates storing all the Fock and density matrices for each root in GPU memory, which becomes problematic for large-scale calculations. The construction of the TDDFT Fock matrix additionally suffers from a larger computational overhead originating from the non-symmetric density matrix.

Full-scale TDDFT on biological protein

As depicted in Fig. 5, we report the full-scale excited-state calculations of green fluorescent protein (GFP). GFP is found in the jellyfish *Aequorea victoria*, and contain *p*-hydroxybenzylidene-imidazolinone (HBI) chromophore within the 11-stranded *β*-barrel as the component responsible for the green fluorescence observed in this species. The protein chain comprises 238 amino acids, and HBI emits bright green fluorescence when exposed to light in the blue to ultraviolet range[68]. The molecular structure was generated from the X-ray crystallography followed by protonation to a pH of 7.8, rendering a total of 4353 atoms, including 245 water molecules, and characterized with −6 charge (Fig. 5a). Solvent water molecules are essential for yielding a non-vanishing energy gap between the highest-occupied and lowest-unoccupied MOs (HOMO–LUMO)[69,70]. The molecular geometry was adopted from that reported in[71] with geometrical corrections on eight amide H atoms. The corrections were made on H atoms that were farther than 1.5 Å from the neighboring N atoms by re-attaching them at 1.0 Å from N. It was originally intended for hybrid quantum mechanics/molecular mechanics (QM/MM) simulations such that only the 42-atom HBI chromophore was treated quantum-mechanically, enabling the embedding DFT calculations within cruder force-field calculations. Thus, this whole GFP had been considered plainly formidable in DFT formalism prior to this work. In the full-scale DFT treatment, the total number of basis functions was $N = 40{,}518$ with the def2-SVP basis set, and the numbers of occupied and virtual orbitals were 8164 and 32,354, respectively. The number of shell-pairs are provided in Supplementary Table 5, showing that < 5% of shell-pairs are retained after the integral-screening as an indication of approaching the asymptotic $\mathcal{O}(N^2)$ behavior of ERIs. However, we note that even with the significantly reduced number of shell-pairs here, the associated ERI computations still overwhelms the linear algebra operations (Supplementary Table 6).

Using our multi-GPU implementation, we have successfully pioneered the ground-state DFT and the excited-state TDDFT calculations for the whole GFP system. We employed the HFLYP

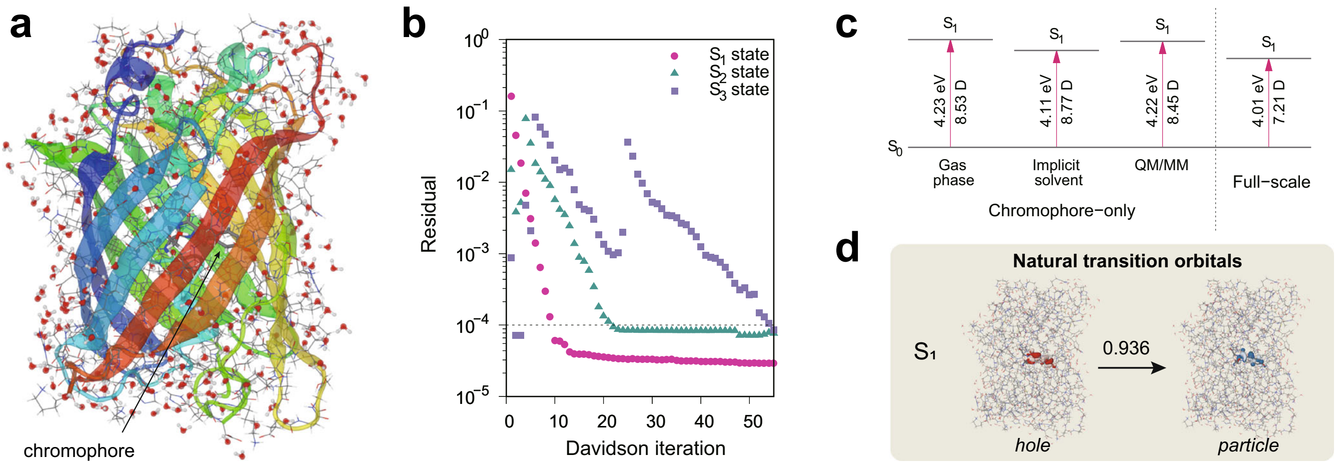


Fig. 5 Excited state of green fluorescent protein (GFP). **a** Molecular structure of GFP complex with surrounding water molecules. Ribbon diagram of GFP is overlaid for visual aid. The chromophore is shown in the center of the *β*-barrel. Total number of atoms is 4353. **b** TDDFT convergence behaviors of the excited states during the Davidson iterations. The convergence threshold is marked with the dotted horizontal line. **c** Bright excitation of the chromophore in various conditions and full-scale GFP complex calculated at the HFLYP/def2-SVP level. Transition dipole moments were also calculated at the same level. **d** Hole/particle wave functions for the S_1 state of GFP are indicated in red/blue as represented by the natural transition orbitals.

exchange–correlation functional, where the exact HF exchange ($c_X = 1.0$) is combined with the Lee–Yang–Parr correlation functional[66,72]. We note that the hybrid functionals with a lower HF exchange either failed to reach SCF convergence due to the closure of the HOMO–LUMO gap or resulted in an unphysically small HOMO–LUMP gap, giving rise to spurious low-lying charge-transfer (CT) states that hampered the excited-state calculations (Supplementary Table 7). The convergence difficulties in the DFT calculations of proteins are generally ascribed to the self-interaction errors within the density functional approximations and the improper treatment of vacuum/system interface, and the viable remedies have been suggested including the use of range-separated functional and implicit solvent model[70,73,74]. Adding extended solvent network as point charges for treating solvated systems, or even truncating the excitation space within the TDDFT description have also been suggested for avoiding contamination by spurious CT states[75]. In this work, where we focus on algorithms for achieving scalable GPU utilizations, we have avoided the difficulty by taking the explicit solvent water molecules into account and simply by using 100% HF exchange.

Figure 5b illustrates the convergence behavior of the lowest three excited-states in the singlet manifold. For all states considered, the convergence threshold of 10^{-4} au for the residual is attained within 55 iterations, entailing a total of 88 $\boldsymbol{\sigma}$-vector formations. The results demonstrate a smooth convergence, and the *bright* excited state was identified as the S_1 state, exhibiting a large transition dipole moment of 7.21 Debye.

Figure 5c demonstrates the variation in excitation energies and transition dipole moments between the HBI chromophore under various model solvent conditions and the entire GFP system. The gas-phase, implicit solvent, and QM/MM models at the same HFLYP level did not capture the inclusion of the protein chains and the explicit water solvent under the experimental pH condition lowered the excitation energy by 0.22 eV (4.23 eV vs 4.01 eV). The excitation depends on the dielectric medium exerted by the protein backbone, and no solvent models can correctly estimate the energy. Similar findings were reported previously that distant residues indeed have an effect on the excitation by using the QM/MM model with varying coverage of the polarizable embedding potential[76]. The high values of the transition dipole moments obtained in the calculations of the HBI chromophore and full-scale GFP system imply strong absorption; however, the latter case exhibits slightly decreased value. Furturemore, a visual inspection of the results of the natural transition orbital analysis[77] revealed that the different calculations resulted in the similar electronic structure centralized in the chromophore (Fig. 5d and Supplementary Fig. 3).

In contrast, in full-scale TDDFT at the B3LYP level, the S_1 state was determined to be the spurious charge-transfer state between the chromophore and a distant residue, and its transition dipole moment was accordingly very small (Supplementary Fig. 3). This is a common problem of conventional global DFT functionals[75], and can be remedied by adopting the range-separation technique. Indeed, when we adopted the LC-*ω*PBE functional, we obtained a larger HOMO–LUMO gap (Supplementary Table 7) and the bright state was correctly predicted as S_1 with an excitation energy of 3.88 eV. In addition, the state was not corroded by the artificial intensity borrowing of spurious low-lying CT states[75]. In light of the strong local excitation characteristics in the bright state of GFP, the influence of diffuse basis functions on the excitation of a neutral system will likely be small. However, actually assessing it in a quantitative manner especially for a large system can be an interesting study in a future work. Of course, they can play an important role in anionic systems, further warranting a future study with diffuse functions.

Finally, Fig. 6a characterizes the multi-GPU performance of the TDDFT calculations of GFP over and up to 256 A100 GPUs totaling 2.5 peta-FLOPS in the raw computing power with double precision. Parallel efficiency is evaluated as the ratio of total computation times on single-GPU and multi-GPUs, describing losses due to communication overhead and serial fractions of the code, as well as the load-balance, and indicating the degree of the actual speed-up with respect to the ideal speed-up. A near perfect parallel efficiency can be expected from the carefully designed dynamic load-balancing scheme described in an earlier section. Using 64 GPUs distributed over 8 compute nodes, a favorable speed-up of 45.5 was achieved, corresponding to a parallel efficiency of 0.71. The parallel efficiency decreased to 0.54 and 0.37 for 128 and 256 GPUs, respectively; this trend resulted, not from load-imbalance but from the emerging serial and partly parallelized sections of the code, owing to the relatively short execution of the GPU kernels from the fine-graining. As observed in Fig. 6b, the total TDDFT execution time decreases substantially from 22.2 days using 1 GPU to a mere 5.6 hours using 256 GPUs. Examining the individual timing components revealed that the high parallel efficiency of ERI and XC computations in the Fock

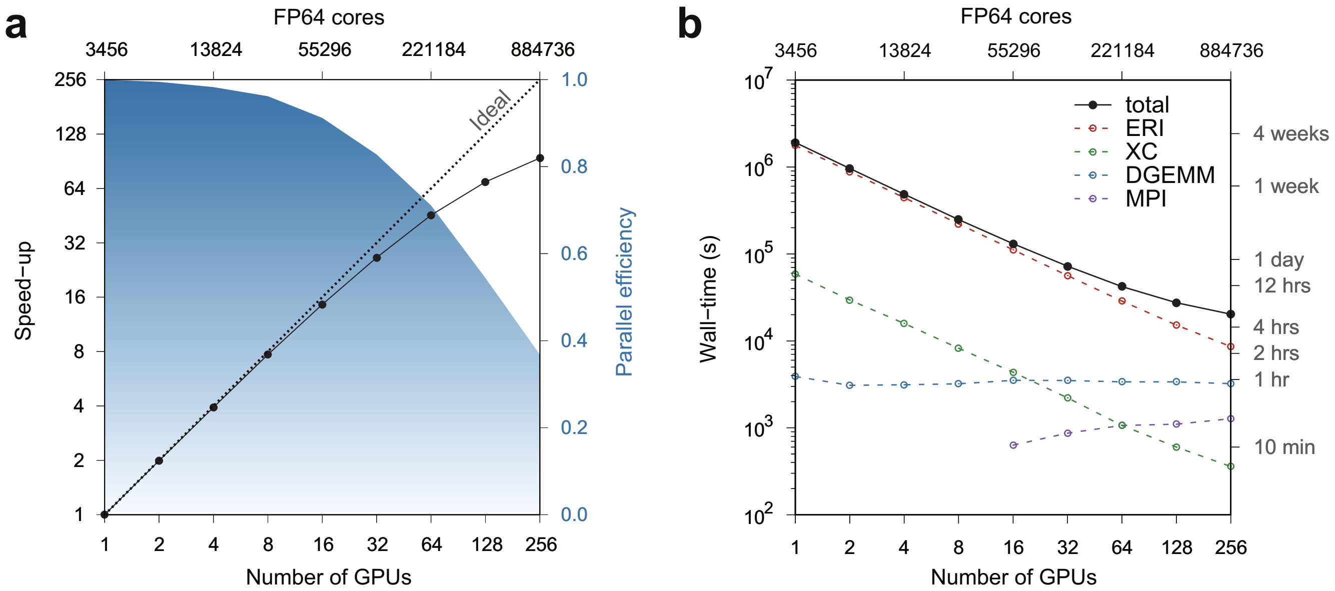


Fig. 6 GPU performance on TDDFT calculations of GFP. a Speed-up and parallel efficiency per number of GPUs. The ideal speed-up is indicated by the dotted diagonal line. The number of GPU FP64 cores are also provided on top of the plot. A separate linear scale plot is provided in Supplementary Fig. 4. **b** Wall-time to convergence per number of GPUs. Alternate units of time are also provided to aid in interpretation of the results. Decomposition into different computations are also provided.

build stage was maintained throughout the benchmark, as indicated by the linearity in the log-log scale, also suggesting that the latency of GPU kernel launches could be effectively hidden. The deteriorated parallel efficiency is mainly attributed to matrix multiplications, featuring limited GPU parallelization within a single node; we note that the parallel efficiency can be easily increased to 0.46 in combination with a scalable multi-GPU matrix operations. It may also be worthwhile to assess the parallel performance of our implementation under the constant workload per GPU. We have provided such information in Supplementary Text 5 for interested readers.

DISCUSSION

In this study, we developed and implemented high-performance DFT and TDDFT algorithms for material simulations of very large size and scale. Specifically, we have considered the Kohn–Sham formulation with finite, atom-centered Gaussian functions, as adopted in standard quantum chemistry suites of programs. The parallelization capabilities of multi-GPU were investigated to accelerate the Fock matrix build, which was determined to be the maximally time-consuming step, comprising the calculations of the ERIs and XC functional derivatives. Three parallelization models of MPI, OpenMP and CUDA programming models were employed for handling the inter-node, intra-node, and GPU parallelizations, respectively. These models were utilized in tandem for a faster and larger scale of application development. The presented GPU algorithm ensured the concurrency of threads within a GPU kernel, while yielding an desirable load-balancing within the distributed network of GPUs.

We demonstrated the performance of our DFT implementation at large scales through a benchmark study with a custom-built state-of-the-art GPU-cluster system comprising 256 Nvidia A100 GPUs. In principle, this system collectively operated at 2.5 peta-FLOPS with double precision, and scaling feasibly to problems of an extremely large size. The excited-state calculation of the green fluorescent protein complex with 4,353 atoms and the def2-SVP basis set in this study marked the largest TDDFT calculation to date on any computing platform. The calculation was concluded within ~6 h with 256 GPUs, achieving a parallel efficiency of 0.37.

In summary, our contribution not only significantly improves and expands the spectrum of molecular systems to be considered by full quantum-chemical treatment without the possible loss of accuracy associated with cost-reducing approximations, but also empowers materials scientists to seek new designs or new combinations of organic molecules to an extent that has remained inconceivable till now.

We note that even with the capability of performing large scale (TD)DFT calculations, more affordable QM/MM style calculations will still be very useful and should be employed in a complementary manner. In fact, from the deterioration of DFT results on full GFP with the widely adopted B3LYP functional, one can infer that treating a large enough system with full-scale DFT does not necessarily gain over using QM/MM or some other multiscale approaches. The HFLYP approach that we employed was an easy way of circumventing the problem caused by the locality issue of DFT and the related appearances of spurious CT states, but will not be a physically acceptable solution as HFLYP is not likely very reliable in handling diverse chemistry problems. The range-separation techniques may contribute again as a more viable tool for handling extended systems, and we will need further tests with their benchmarks in that regard. In addition, while we observed ~0.2 eV shift in the excitation energy with the inclusion of the full protein model with GFP, how far in space we should extend from the chromophore itself to reach some convergence is still a question that needs to be answered. Of course, research for answering these should be designed also by considering the form of the adopted exchange–correlation functional and perhaps with different levels of basis sets. We anticipate that such studies will ensue in the near future, and QM/MM will definitely be utilized for useful comparisons. As these will involve heavy computations, a method as reported in this work that can utilize a highly parallel GPU platform will be extremely helpful.

METHODS

Code implementation

The multi-GPU DFT algorithms were implemented in an in-house code evolved from[78]. The program was written entirely in modern

Fortran 2003/2008 with the CUDA extension, and was compiled using Nvidia SDK 22.3 compiler suite with the "-fast" optimization flag. The linear algebra library of Nvidia's cuBLAS 11.8 and cuSOLVER 11.3 were utilized for matrix multiplications, diagonalizations, and singular value decompositions. The multithreaded version of the Mellanox HPC-X 2.10 package was employed for MPI based on OpenMPI 4.1.2. Unified communication-X (UCX) was leveraged for both point-to-point and one-sided communication between the nodes. CPU-to-GPU binding was achieved using OpenMP parallel interfaces. We considered a non-uniform memory access (NUMA) structure in a multithreaded hybrid OpenMP/MPI application for maximizing CPU-to-GPU and node-to-node communications and data transfers. This enhancement was achieved effortlessly within the OpenMP environment by enabling the `OMP_PROC_BIND` variable and explicitly binding the cores in the `OMP_PLACES` variable according to the hardware configurations (refer to Supplementary Fig. 5 and Supplementary Table 8).

Electronic structure calculations

Multithreaded DFT and TDDFT calculations on the CPU were performed using the Turbomole 7.6 program package[63]. The solvent effect in the excited state was investigated using a continuum solvation model, the conductor-like screening model (COSMO)[79,80]. The aqueous conditions were imitated using the parameters of dielectric constant ($\epsilon = 78.35$) and refractive index ($n = 1.3$). The QM/MM calculations were performed using the Gaussian 16.C.01 program[81] with the our-own-N-layer integrated molecular orbital molecular mechanics (ONIOM) approach[82] with electrostatic embedding[83], in which the Amber force fields were used to describe the MM region. A conventional unpruned (50, 194)-grid was employed for numerical integration (i.e., 50 radial points and 194 angular points per radial point without any pruning)[84], while the implementation-default XC grids were used with Turbomole and Gaussian. Further, the threshold of the SCF convergence was the root-mean-squared difference of 10^{-6} au in two consecutive density matrices. A symmetric orthonormalization procedure with the threshold of 10^{-6} au was used to define the orthonormal orbitals. For the range-separated LC-ωPBE functional, for which we have not implemented efficient GPU treatments on the XC energies, we evaluated potentials and kernels by using the CPU version of LibXC 5.2.3[85]. Natural transition orbital (NTO) analyses for the CPU calculations were post-processed using the TheoDORE 2.0 package[86]. The def2-SVP basis set was used for all calculations[87]. All molecular geometries considered in this work are provided in Supplementary Table 9.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding authors upon reasonable request.

CODE AVAILABILITY

The computer code developed within this work is proprietary and its copyright belongs to Samsung Electronics.

Received: 4 January 2023; Accepted: 8 May 2023;
Published online: 26 May 2023

REFERENCES

1. Louie, S. G., Chan, Y.-H., da Jornada, F. H., Li, Z. & Qiu, D. Y. Discovering and understanding materials through computation. *Nat. Mater.* **20**, 728–735 (2021).
2. Hohenberg, P. & Kohn, W. Inhomogeneous electron gas. *Phys. Rev. B* **136**, 864–871 (1964).
3. Kohn, W. & Sham, L. J. Self-consistent equations including exchange and correlation effects. *Phys. Rev.* **140**, A1133–A1138 (1965).
4. Mardirossian, N. & Head-Gordon, M. Thirty years of density functional theory in computational chemistry: an overview and extensive assessment of 200 density functionals. *Mol. Phys.* **115**, 2315–2372 (2017).
5. Herbert, J. M. Density functional theory for electronic excited states. In *Theoretical and computational photochemistry* (eds. Cristina, G. I. & Marazzi, M.) 69–118 (Elsevier, 2023).
6. Burke, K. Perspective on density functional theory. *J. Chem. Phys.* **136**, 150901 (2012).
7. Becke, A. D. Perspective: fifty years of density-functional theory in chemical physics. *J. Chem. Phys.* **140**, 18A301 (2014).
8. Bursch, M., Mewes, J., Hansen, A. & Grimme, S. Best practice DFT protocols for basic molecular computational chemistry. *Angew. Chem. Int. Ed.* **61**, e202205735 (2022).
9. Kussmann, J., Beer, M. & Ochsenfeld, C. Linear-scaling self-consistent field methods for large molecules. *WIREs Comput. Mol. Sci.* **3**, 614–636 (2013).
10. Heldens, S. et al. The landscape of exascale research. *ACM Comput. Surv.* **53**, 1–43 (2021).
11. McInnes, L. C. et al. How community software ecosystems can unlock the potential of exascale computing. *Nat. Comput. Sci.* **1**, 92–94 (2021).
12. Kirk, D. Nvidia CUDA software and GPU parallel computing architecture. In *Proceedings of the 6th International Symposium on Memory Management*, ISMM '07, 103–104 (Association for Computing Machinery, 2007).
13. Seritan, S. et al. TeraChem: A graphical processing unit-accelerated electronic structure package for large-scale ab initio molecular dynamics. *WIREs Comput. Mol. Sci.* **11**, e1494 (2021).
14. Ufimtsev, I. S. & Martínez, T. J. Quantum chemistry on graphical processing units. 1. Strategies for two-electron integral evaluation. *J. Chem. Theory Comput.* **4**, 222–231 (2008).
15. Yasuda, K. Two-electron integral evaluation on the graphics processor unit. *J. Comput. Chem.* **29**, 334–342 (2008).
16. Yasuda, K. Accelerating density functional calculations with graphics processing unit. *J. Chem. Theory Comput.* **4**, 1230–1236 (2008).
17. Nagy, B. & Jensen, F. *Basis sets in quantum chemistry, Chap. 3* (Wiley, 2017).
18. Hacene, M. et al. Accelerating vasp electronic structure calculations using graphic processing units. *J. Comput. Chem.* **33**, 2581–2589 (2012).
19. Andrade, X. & Aspuru-Guzik, A. Real-space density functional theory on graphical processing units: computational approach and comparison to gaussian basis set methods. *J. Chem. Theory Comput.* **9**, 4360–4373 (2013).
20. Asadchev, A. et al. Uncontracted Rys quadrature implementation of up to g functions on graphical processing units. *J. Chem. Theory Comput.* **6**, 696–704 (2010).
21. Wilkinson, K. A., Sherwood, P., Guest, M. F. & Naidoo, K. J. Acceleration of the GAMESS-UK electronic structure package on graphical processing units. *J. Comput. Chem.* **32**, 2313–2318 (2011).
22. Titov, A. V., Ufimtsev, I. S., Luehr, N. & Martínez, T. J. Generating efficient quantum chemistry codes for novel architectures. *J. Chem. Theory Comput.* **9**, 213–221 (2013).
23. Miao, Y. & Merz, K. M. Acceleration of electron repulsion integral evaluation on graphics processing units via use of recurrence relations. *J. Chem. Theory Comput.* **9**, 965–976 (2013).
24. Rák, A. & Cserey, G. The BRUSH algorithm for two-electron integrals on GPU. *Chem. Phys. Lett.* **622**, 92–98 (2015).
25. Kussmann, J. & Ochsenfeld, C. Hybrid CPU/GPU integral engine for strong-scaling ab initio methods. *J. Chem. Theory Comput.* **13**, 3153–3159 (2017).
26. Kalinowski, J., Wennmohs, F. & Neese, F. Arbitrary angular momentum electron repulsion integrals with graphical processing units: Application to the resolution of identity Hartree–Fock method. *J. Chem. Theory Comput.* **13**, 3160–3170 (2017).
27. Tornai, G. J., Ladjánszki, I., Ádám, R., Kis, G. & Cserey, G. Calculation of quantum chemical two-electron integrals by applying compiler technology on GPU. *J. Chem. Theory Comput.* **15**, 5319–5331 (2019).
28. Ufimtsev, I. S. & Martínez, T. J. Quantum chemistry on graphical processing units. 2. Direct self-consistent-field implementation. *J. Chem. Theory Comput.* **5**, 1004–1015 (2009).
29. Miao, Y. & Merz, K. M. Acceleration of high angular momentum electron repulsion integrals and integral derivatives on graphics processing units. *J. Chem. Theory Comput.* **11**, 1449–1462 (2015).
30. Barca, G. M. J., Galvez-Vallejo, J. L., Poole, D. L., Rendell, A. P. & Gordon, M. S. High-performance, graphics processing unit-accelerated fock build algorithm. *J. Chem. Theory Comput.* **16**, 7232–7238 (2020).
31. Kussmann, J. & Ochsenfeld, C. Pre-selective screening for matrix elements in linear-scaling exact exchange calculations. *J. Chem. Phys.* **138**, 134114 (2013).
32. Kussmann, J. & Ochsenfeld, C. Preselective screening for linear-scaling exact exchange-gradient calculations for graphics processing units and general strong-

scaling massively parallel calculations. *J. Chem. Theory Comput.* **11**, 918–922 (2015).

33. Manathunga, M., Miao, Y., Mu, D., Götz, A. W. & Merz, K. M. Parallel implementation of density functional theory methods in the quantum interaction computational kernel program. *J. Chem. Theory Comput.* **16**, 4315–4326 (2020).
34. Williams-Young, D. B., de Jong, W. A., van Dam, H. J. J. & Yang, C. On the efficient evaluation of the exchange correlation potential on graphics processing unit clusters. *Front. Chem.* **8**, 581058 (2020).
35. Seritan, S. et al. TeraChem: accelerating electronic structure and ab initio molecular dynamics with graphical processing units. *J. Chem. Phys.* **152**, 224110 (2020).
36. Manathunga, M. et al. Harnessing the power of multi-GPU acceleration into the quantum interaction computational kernel program. *J. Chem. Theory Comput.* **17**, 3955–3966 (2021).
37. Barca, G. M. J. et al. Faster self-consistent field (SCF) calculations on GPU clusters. *J. Chem. Theory Comput.* **17**, 7486–7503 (2021).
38. Isborn, C. M., Luehr, N., Ufimtsev, I. S. & Martínez, T. J. Excited-state electronic structure with configuration interaction singles and Tamm–Dancoff time-dependent density functional theory on graphical processing units. *J. Chem. Theory Comput.* **7**, 1814–1823 (2011).
39. CUDA C++ Programming Guide, Nvidia. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.
40. The SSC-21 supercomputer has been ranked at 15th on the Top500 list of supercomputers. https://www.top500.org/lists/top500/list/2022/06/.
41. Perdew, J. P., Burke, K. & Ernzerhof, M. Generalized gradient approximation made simple. *Phys. Rev. Lett.* **77**, 3865–3868 (1996).
42. Boys, S. F. Electronic wave functions – I. A general method of calculation for the stationary states of any molecular system. *Proc. R. Soc. Lond. A: Math. Phys. Sci.* **200**, 542–554 (1950).
43. Roothaan, C. C. J. New developments in molecular orbital theory. *Rev. Mod. Phys.* **23**, 69–89 (1951).
44. Hall, G. G. The molecular orbital theory of chemical valency VIII. A method of calculating ionization potentials. *Proc. R. Soc. Lond. A Math. Phys. Sci.* **205**, 541–552 (1951).
45. Pople, J. A., Gill, P. M. & Johnson, B. G. Kohn–Sham density-functional theory within a finite basis set. *Chem. Phys. Lett.* **199**, 557–560 (1992).
46. Runge, E. & Gross, E. K. U. Density-functional theory for time-dependent systems. *Phys. Rev. Lett.* **52**, 997–1000 (1984).
47. Chong, D. P. (ed.) *Recent advances in density functional methods* (World Scientific Publishing Co. Pte. Ltd., 1995).
48. Hirata, S. & Head-Gordon, M. Time-dependent density functional theory within the Tamm–Dancoff approximation. *Chem. Phys. Lett.* **314**, 291–299 (1999).
49. Casida, M. & Huix-Rotllant, M. Progress in time-dependent density-functional theory. *Annu. Rev. Phys. Chem.* **63**, 287–323 (2012).
50. Chantzis, A., Laurent, A. D., Adamo, C. & Jacquemin, D. Is the Tamm-Dancoff approximation reliable for the calculation of absorption and fluorescence band shapes? *J. Chem. Theory Comput.* **9**, 4517–4525 (2013).
51. Peach, M. J. G., Williamson, M. J. & Tozer, D. J. Influence of triplet instabilities in TDDFT. *J. Chem. Theory Comput.* **7**, 3578–3585 (2011).
52. Davidson, E. R. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *J. Comput. Phys.* **17**, 87–94 (1975).
53. Leininger, M. L., Sherrill, C. D., Allen, W. D. & Schaefer, H. F. Systematic study of selected diagonalization methods for configuration interaction matrices. *J. Comput. Chem.* **22**, 1574–1589 (2001).
54. Weiss, H., Ahlrichs, R. & Häser, M. A direct algorithm for self-consistent-field linear response theory and application to C_{60} : excitation energies, oscillator strengths, and frequency-dependent polarizabilities. *J. Chem. Phys.* **99**, 1262–1270 (1993).
55. Almlöf, J., Faegri, K. & Korsell, K. Principles for a direct SCF approach to LCAO–MO ab-initio calculations. *J. Comput. Chem.* **3**, 385–399 (1982).
56. McMurchie, L. E. & Davidson, E. R. One- and two-electron integrals over cartesian Gaussian functions. *J. Comput. Phys.* **26**, 218–231 (1978).
57. Head-Gordon, M. & Pople, J. A. A method for two-electron gaussian integral and integral derivative evaluation using recurrence relations. *J. Chem. Phys.* **89**, 5777–5786 (1988).
58. Dupuis, M., Rys, J. & King, H. F. Evaluation of molecular integrals over Gaussian basis functions. *J. Chem. Phys.* **65**, 111–116 (1976).
59. Häser, M. & Ahlrichs, R. Improvements on the direct SCF method. *J. Comput. Chem.* **10**, 104–111 (1989).
60. Becke, A. D. A multicenter numerical integration scheme for polyatomic molecules. *J. Chem. Phys.* **88**, 2547–2553 (1988).
61. Murray, C. W., Handy, N. C. & Laming, G. J. Quadrature schemes for integrals of density functional theory. *Mol. Phys.* **78**, 997–1014 (1993).
62. Lebedev, V. I. Spherical quadrature formulas exact to orders 25–29. *Sib. Math. J.* **18**, 99–107 (1977).
63. Balasubramani, S. G. et al. Turbomole: modular program suite for ab initio quantum-chemical and condensed-matter simulations. *J. Chem. Phys.* **152**, 184107 (2020).
64. Zhang, D., Cai, M., Zhang, Y., Zhang, D. & Duan, L. Sterically shielded blue thermally activated delayed fluorescence emitters with improved efficiency and stability. *Mater. Horiz.* **3**, 145–151 (2016).
65. Becke, A. D. Density-functional thermochemistry. III. The role of exact exchange. *J. Chem. Phys.* **98**, 5648–5652 (1993).
66. Lee, C., Yang, W. & Parr, R. G. Development of the Colle–Salvetti correlation-energy formula into a functional of the electron density. *Phys. Rev. B* **37**, 785–789 (1988).
67. Stephens, P. J., Devlin, F. J., Chabalowski, C. F. & Frisch, M. J. Ab initio calculation of vibrational absorption and circular dichroism spectra using density functional force fields. *J. Phys. Chem.* **98**, 11623–11627 (1994).
68. Zimmer, M. Green fluorescent protein (GFP): applications, structure, and related photophysical behavior. *Chem. Rev.* **102**, 759–782 (2002).
69. Rudberg, E. Difficulties in applying pure Kohn–Sham density functional theory electronic structure methods to protein molecules. *J. Phys. Condens. Matter* **24**, 072202 (2012).
70. Lever, G., Cole, D. J., Hine, N. D. M., Haynes, P. D. & Payne, M. C. Electrostatic considerations affecting the calculated HOMO-LUMO gap in protein molecules. *J. Phys. Condens. Matter* **25**, 152101 (2013).
71. Foresman, J. & Frisch, Æ. *Exploring chemistry with electronic structure methods* 3rd edn (Gaussian, Inc., 2015),
72. Miehlich, B., Savin, A., Stoll, H. & Preuss, H. Results obtained with the correlation energy density functionals of Becke and Lee, Yang and Parr. *Chem. Phys. Lett.* **157**, 200–206 (1989).
73. Kulik, H. J., Luehr, N., Ufimtsev, I. S. & Martínez, T. J. Ab initio quantum chemistry for protein structures. *J. Phys. Chem. B* **116**, 12501–12509 (2012).
74. Antony, J. & Grimme, S. Fully ab initio protein-ligand interaction energies with dispersion corrected density functional theory. *J. Comput. Chem.* **33**, 1730–1739 (2012).
75. Lange, A. & Herbert, J. M. Simple methods to reduce charge-transfer contamination in time-dependent density-functional calculations of clusters and liquids. *J. Chem. Theory Comput.* **3**, 1680–1690 (2007).
76. Schwabe, T., Beerepoot, M. T. P., Olsen, J. M. H. & Kongsted, J. Analysis of computational models for an accurate study of electronic excitations in GFP. *Phys. Chem. Chem. Phys.* **17**, 2582–2588 (2015).
77. Martin, R. L. Natural transition orbitals. *J. Chem. Phys.* **118**, 4775 (2003).
78. Kim, I. & Lee, Y. S. KPACK: Relativistic two-component ab initio electronic structure program package. *Bull. Korean Chem. Soc.* **34**, 179–187 (2013).
79. Klamt, A. & Schüürmann, G. COSMO: a new approach to dielectric screening in solvents with explicit expressions for the screening energy and its gradient. *J. Chem. Soc. Perkin Trans.* **2**, 799–805 (1993).
80. Klamt, A. Calculation of UV/Vis spectra in solution. *J. Phys. Chem.* **100**, 3349–3353 (1996).
81. Frisch, M. J. et al. *Gaussian 16 Revision C.01*, (Gaussian Inc., 2016).
82. Svensson, M. et al. ONIOM: A multilayered integrated MO + MM method for geometry optimizations and single point energy predictions. A test for Diels-Alder reactions and $Pt(P(t\text{-}Bu)_3)_2 + H_2$ oxidative addition. *J. Phys. Chem.* **100**, 19357–19363 (1996).
83. Bakowies, D. & Thiel, W. Hybrid models for combined quantum mechanical and molecular mechanical approaches. *J. Phys. Chem.* **100**, 10580–10594 (1996).
84. Gill, P. M., Johnson, B. G. & Pople, J. A. A standard grid for density functional calculations. *Chem. Phys. Lett.* **209**, 506–512 (1993).
85. Lehtola, S., Steigemann, C., Oliveira, M. J. & Marques, M. A. Recent developments in libxc – a comprehensive library of functionals for density functional theory. *SoftwareX* **7**, 1–5 (2018).
86. Plasser, F. TheoDORE: a toolbox for a detailed and automated analysis of electronic excited state computations. *J. Chem. Phys.* **152**, 084108 (2020).
87. Weigend, F. & Ahlrichs, R. Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: design and assessment of accuracy. *Phys. Chem. Chem. Phys.* **7**, 3297 (2005).

ACKNOWLEDGEMENTS

The authors thank Prof. Yoon Sup Lee (KAIST) for helpful discussions. I.K. thanks Dr. Tae-Rae Kim (Samsung Electronics) for assisting with code reviews. Y.M.R. acknowledges Samsung Electronics for financial support. This work was in part supported by the National Research Foundation (NRF) of Korea (Grant No. 2020R1A5A1019141 and 2021R1A2C2094153). Computational resources were provided by the Supercomputing Center of Samsung Electronics.

AUTHOR CONTRIBUTIONS

I.K., W.-J.S., H.-J.K. and Y.M.R. conceived and designed the project and wrote the manuscript. I.K. developed theoretical models and wrote the code. D.J. performed quantum chemistry calculations. Y.J. and H.C. contributed to the interpretation of results. J.Y. assisted with hardware set-up and GPU performance profiling. I.J. and D.S.K. organized the project and supervised the computational study. All authors analyzed the data, discussed the results, and contributed to various portions of the manuscript.

COMPETING INTERESTS

The authors declare no competing interests.

ADDITIONAL INFORMATION

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41524-023-01041-4.

Correspondence and requests for materials should be addressed to Won-Joon Son, Hyung-Jin Kim or Young Min Rhee.

Reprints and permission information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.