Parallel Computing 120 (2024) 103085

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Extending the limit of LR-TDDFT on two different approaches: Numerical algorithms and new Sunway heterogeneous supercomputer ☆

Qingcai Jiang [a,1], Zhenwei Cao [a,1], Xinhui Cui [a], Lingyun Wan [a], Xinming Qin [a], Huanqi Cao [b], Hong An [a,*], Junshi Chen [a,*], Jie Liu [a], Wei Hu [a,*], Jinlong Yang [a]

[a] *University of Science and Technology of China, Hefei, China*
[b] *Department of Computer Science and Technology & BNRist, Tsinghua University, Beijing, China*

ARTICLE INFO

Keywords:
Quantum mechanics
Time-dependent density functional theory
Linear-response
Iterative eigensolver
Low-rank approximation
Parallel implementation

ABSTRACT

First-principles time-dependent density functional theory (TDDFT) is a powerful tool to accurately describe the excited-state properties of molecules and solids in condensed matter physics, computational chemistry, and materials science. However, a perceived drawback in TDDFT calculations is its ultrahigh computational cost $\mathcal{O}(N^5 \sim N^6)$ and large memory usage $\mathcal{O}(N^4)$ especially for plane-wave basis set, confining its applications to large systems containing thousands of atoms. Here, we present a massively parallel implementation of linear-response TDDFT (LR-TDDFT) and accelerate LR-TDDFT in two different aspects: (1) numerical algorithms on the X86 supercomputer and (2) optimizations on the heterogeneous architecture of the new Sunway supercomputer. Furthermore, we carefully design the parallel data and task distribution schemes to accommodate the physical nature of different computation steps. By utilizing these two different methods, our implementation can gain an overall speedup of 10x and 80x and efficiently scales to large systems up to 4096 and 2744 atoms within dozens of seconds.

1. Introduction

First-principles Kohn–Sham density functional theory (DFT) [2] has extensive applications in condensed matter physics, computational chemistry and materials science. To enable the computation-based design of new materials and predict their peculiar properties in different types of fields with high accuracy, developing large-scale DFT and time-dependent DFT (TDDFT) [3] methods both in ground-state and excited-state simulations is of significant impact.

In particular, TDDFT, as one of the most widely used and powerful computational tools for exploring excited-state properties, has inspired a variety of imaginative methods designed for various purposes. Among them, real-time TDDFT [4–6] focuses on dynamic evolution processes in the real-time domain, offering a unique ab-initio method applicable for exploring particle interactions, especially under strong fields or in various spectral regions. However, it comes with a higher computational cost. Therefore, with the assistance of perturbation theory, many methods opt to address the problem from the perspective of the frequency domain. For instance, damped response theory [7–9], by solving the standard response equation then allowing for absorption, could directly applies to high-frequency or high-density-of-states spectral regions, making it the best choice for X-ray absorption spectroscopy (XAS) spectra and plasmonic systems. Additionally, the most common formula to adopt excited-state properties, which evaluates exactly the many-body Schrödinger equation of the time-dependent linear response function formulated in the frequency domain, refers to as linear-response TDDFT (LR-TDDFT).

Within LR-TDDFT, the Casida equation [10] is the most commonly used formula to describe the excitation energy and corresponding wavefunctions. To solve the Casida equation, the most time-consuming parts in the LR-TDDFT calculations can be summarized as two parts, one is to explicitly construct the LR-TDDFT Hamiltonian with the complexity of $\mathcal{O}(N_e^5)$ with respect to the number of electrons in the system N_e, and the other is to diagonalize the LR-TDDFT Hamiltonian with the complexity of $\mathcal{O}(N_e^6)$. As the system expands, the computational and memory

☆ This paper is a conference extension article of 51st International Conference on Parallel Processing (ICPP) (Jiang et al., 2022 [1]). The extended part specifically focuses on the optimization and acceleration on the new Sunway heterogeneous platform.

* Corresponding authors.

E-mail addresses: jqc@mail.ustc.edu.cn (Q. Jiang), caozhenwei@mail.ustc.edu.cn (Z. Cao), cuixinhui@mail.ustc.edu.cn (X. Cui), wanly@mail.ustc.edu.cn (L. Wan), xmqin03@ustc.edu.cn (X. Qin), caohq18@mails.tsinghua.edu.cn (H. Cao), han@ustc.edu.cn (H. An), cjuns@ustc.edu.cn (J. Chen), liujie86@ustc.edu.cn (J. Liu), whuustc@ustc.edu.cn (W. Hu), jlyang@ustc.edu.cn (J. Yang).

[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.parco.2024.103085
Received 28 June 2023; Received in revised form 20 April 2024; Accepted 30 April 2024
Available online 4 May 2024

cost of LR-TDDFT calculations in a general CPU platform becomes prohibitively expensive, especially on large complete basis sets, such as plane-wave basis sets. Therefore, exploring the excited-state properties of systems with thousands of atoms using the LR-TDDFT method is still a very tough task.

The state-of-the-art LR-TDDFT calculations in CP2K with Gaussian-type orbitals (GTOs) allow the study of large-scale systems including aluminosilicate imogolite nanotubes, in addition to surface and bulk vacancy defects in MgO and HfO_2 with nearly 1000 atoms [11]. However, the above software is all implemented under localized basis sets, which are not commensurate with the desired accuracy especially when the system is complex. In particular, there has been no breakthrough for a long time with regard to the standard plane-wave basis set because of ultra-high computational and memory cost in the LR-TDDFT calculations, which hinders excited-state electronic structure exploration for large-scale periodic systems containing thousands of atoms.

Fortunately, this situation can be immensely improved thanks to the appearance of new algorithmic methods and modern high performance computing (HPC) facilities. For example, low-rank methods like density fitting approximation, also known as the resolution of identity algorithms [12], can help not only accelerate the construction of LR-TDDFT Hamiltonian but also significantly lower the memory cost. Furthermore, the iterative subspace eigensolver algorithms, such as Davidson [13] and LOBPCG [14], have been successfully applied to simulate excited-state properties by giving an estimation of the lowest k eigenvalues with a favorable computational cost of $\mathcal{O}(kN_e^4)$ (k is the number of desired lowest eigenvalues and corresponding eigenvectors). Given all these advances, the large-scale excited-state calculations of the LR-TDDFT framework with plane-wave basis set become reachable.

In this work, we present a massively parallel implementation of linear-response TDDFT (LR-TDDFT) and accelerate LR-TDDFT in two **different** aspects in the following:

- **Numerical algorithms** such as K-Means based parallel interpolative separable density fitting (ISDF) [1,15], which provides us an efficient and accurate way to reduce the ultra-high computational and memory cost during the construction of Hamiltonian in the simulation of LR-TDDFT. Also, we reduce the computational and memory cost by implicitly constructing and iteratively diagonalizing the Hamiltonian.
- **Heterogeneous architecture of the new Sunway supercomputer** that harnesses the power of numerous computing processor units (CPEs) to deliver remarkable computational capabilities.

Moreover, we perform extensive numerical experiments on the Cori supercomputer, the Cray XC40 system in the National Energy Research Scientific Computing Center (NERSC), and the new Sunway supercomputer, the successor of the Sunway TaihuLight supercomputer.

The main contributions of this work can be summarized as follows:

(1) A series of parallel algorithms, including K-Means based low-rank decomposition, iterative eigenvalue solver and implicit Hamiltonian method are implemented to reduce the computation and memory cost, expand the system size and accelerate the computation steps in the LR-TDDFT calculations.
(2) We present a highly efficient implementation of LR-TDDFT on the new Sunway supercomputer. Our approach incorporates innovative techniques that effectively utilize the architecture of this heterogeneous system, resulting in significant speedup for LR-TDDFT.
(3) We demonstrate that with our algorithms along with parallel implementations and optimizations, we can study the three-dimensional semiconducting silicon systems with 4096 atoms, this result exceeds the current state of the art both in parallel scale and the system scale.
(4) Under extensive experiments, we show that our methods can achieve high scalability, with regard to both strong scaling and weak scaling.
(5) We have open-sourced our software at https://bitbucket.org/berkeleylab/scales/src/lrtddft/ https://bitbucket.org/ berkeley-lab/scales/src/lrtddft/ in the hope that our approach can provide insight for relevant high-performance applications with the same computational characteristics.

2. Related works

Although different types of basis sets can be used in the DFT and TDDFT calculations, plane-wave (PW) basis set [16] in the broadest sense seems current to be the most advantageous for complex periodic solid systems in condensed matter physics and materials science, compared to small localized atomic orbitals (AO) [4] basis set, which is more suitable for molecular systems in quantum chemistry. In particular, PW basis set is complete and allows a faithful analytical evaluation of the total energy, atomic forces, and other physical quantities. But the computational cost of DFT within plane-wave basis set increases rapidly with respect to the number of electrons in the systems because the number of PW basis set is much more expensive than the case of small localized AO basis sets ($N_{\mathrm{PW}} \approx 100 \times N_{\mathrm{AO}}$), which hinders its practical applications to large systems containing thousands of atoms. For example, traditional ground-state DFT calculations with plane-wave basis set are exorbitantly expensive due to $\mathcal{O}(N_e^3)$ scaling computational and memory complexity with respect to the number of electrons $N_e \approx 1000$ [17].

Although it is quite difficult for this software with plane-wave basis to expand to large systems, large-scale TDDFT excited-state electronic structure calculations within small localized basis sets (like atomic and Gaussian basis set) for molecular systems have been implemented recently, such as NWChem [21] and QChem [22]. In detail, as shown in Table 1, NWChem [23] was used to study the excited-state properties of the system containing 120 atoms with 1840 6-311G Gaussian basis set ($Au_{20}Ne_{100}$), and in that work, NWChem efficiently scales to 2250 CPU cores on the CINECA supercomputer.

For periodic solid systems, the GW approximation derived from the Green's function has also become a powerful formalism for studying single-electron excitations of molecules and the quasi-particle band gaps of solids within many-body effects. Recently, large-scale GW calculations containing 2742 atoms within the plane-wave basis set in BerkeleyGW [20] have also been implemented in the Summit supercomputer.

3. Theoretical algorithms of LR-TDDFT

The LR-TDDFT calculations consist of two parts: (1) constructing Hamiltonian and (2) diagonalizing Hamiltonian.

The Hamiltonian we need to construct in LR-TDDFT calculations has the following numerical structure:

$$H = \begin{bmatrix} D + 2V_{\mathrm{Hxc}} & 2W_{\mathrm{Hxc}} \\ -2W_{\mathrm{Hxc}} & -D - 2V_{\mathrm{Hxc}} \end{bmatrix}, \tag{1}$$

where $D\left(i_v i_c, j_v j_c\right) = \left(\varepsilon_{i_c} - \varepsilon_{i_v}\right)\delta_{i_v j_v}\delta_{i_c j_c}$, is an $N_{cv} \times N_{cv}$ ($N_{cv} = N_c \times N_v$, in which N_c is the number of conduction orbitals, N_v is the number of valence orbitals and δ denotes Dirac delta function) matrix. These orbital energies ($\varepsilon_{i_v}(i_v = 1, \ldots, N_v)$ and $\varepsilon_{i_c}(i_c = 1, \ldots, N_c)$) and corresponding orbitals are typically obtained via ground-state Kohn–Sham DFT calculations. The V_{Hxc} and W_{Hxc} matrices represent the Hartree-exchange–correlation integrals.

With the Tamm–Dancoff approximation (TDA) [24], W_{Hxc} matrix is neglectable so the Hamiltonian matrix has the form

$$H = D + 2V_{\mathrm{Hxc}}. \tag{2}$$

In discrete cases, V_{Hxc} is defined as the multiplication of the matrix f_{Hxc} and transposed block face-splitting product (or Block column-wise version of the Khatri–Rao product) matrix $P_{vc} = \{\psi_{i_v}(\mathbf{r})\psi_{i_c}(\mathbf{r})\}$. $\psi_{i_v}(\mathbf{r})$ and $\psi_{i_c}(\mathbf{r})$ stand for the valence and conduction orbitals in real

Table 1
Performance comparison of massively parallel excited-state simulation software packages on modern heterogeneous supercomputers, involving different HPC codes (NWChem, CP2K, PWDFT, and BerkeleyGW) within different types of basis sets (Plane-wave (PW), Gaussian and mixed Gaussian and plane wave (GPW)).

HPC software	Year	Theory	Basis set	Method	System	#atoms	Architecture	Reference
NWChem	2016	LR-TDDFT	Gaussian	Explicit	Water molecules	1,890	Intel Xeon	[18]
CP2K	2019	LR-TDDFT	GPW	Explicit	MgO; HfO_2	1,000	Intel Xeon	[11]
PWDFT	2019	RT-TDDFT	PW	Implicit	Silicon	1,536	V100 GPU	[19]
BerkeleyGW	2020	GW	PW	Explicit	Silicon	2,742	V100 GPU	[20]
PWDFT	2023	LR-TDDFT	PW	Implicit	Silicon; Graphene	4,096	Intel Xeon	This work

Table 2
Computation and memory complexity for constructing and diagonalizing the LR-TDDFT Hamiltonian matrix with the naïve LR-TDDFT code.

LR-TDDFT		Computation	Memory
Constructing Hamiltonian	Face-splitting product of conduction-valence orbitals	$\mathcal{O}(N_v N_c N_r)$	$\mathcal{O}(N_v N_c N_r)$
	Fast fourier transform (FFT)	$\mathcal{O}(N_v^2 N_c^2 N_r)$	$\mathcal{O}(N_v N_c N_r)$
	General matrix multiply (GEMM)	$\mathcal{O}(N_v^2 N_c^2 N_r)$	$\mathcal{O}(N_v^2 N_c^2)$
	f_{Hxc} kernel	$\mathcal{O}(N_v N_c N_r)$	$\mathcal{O}(N_v N_c N_r)$
Diagonalizing Hamiltonian	ScaLAPACK::Syevd	$\mathcal{O}(N_v^3 N_c^3)$	$\mathcal{O}(N_v^2 N_c^2)$

Algorithm 1 The pseudocode for the LR-TDDFT calculations.

```
Input: Ground-state energies $\epsilon_i$, wavefunctions $\psi_\mu(\mathbf{r})$ and $\psi_\nu(\mathbf{r})$ distributed according to the row index.
1:  for each MPI process do
2:      Initialize $P_{vc}(r) = \{\psi_\mu(r)\psi_\nu(r)\}$ in real space;
3:      Carry MPI_Alltoall to wavefunctions $\Psi$ to transfer data distribution scheme from row block partition to column block partition;
4:      Transfer $P_{vc}(g)$ into reciprocal space via fast Fourier transform (FFT);
5:      Apply the Hartree potential operator in reciprocal space and transfer it back into real space $v_H(g)P_{vc}(g)$;
6:      Carry out MPI_Alltoall to wavefunctions $\Psi$ to transfer data distribution scheme from column block partition to row block partition;
7:      Compute the Hartree-exchange-correlation integrals $V_{Hxc}$ in real space via general matrix multiply (GEMM);
8:      Summarize $V_{Hxc}$ within all MPI tasks by MPI_Allreduce;
9:  end for
10: Obtain Hamiltonian by computing the difference of Kohn-Sham energy eigenvalues;
11: Diagonalize the Hamiltonian;
Output: Excited-state energies $\{\lambda_i\}$ and wavefunctions $\{x_{ij}\}$
```

space ($\{\mathbf{r_i}\}_{i=1}^{N_r}$, N_r denotes the number of real space grid points during the calculations).

$$V_{\mathrm{Hxc}} = P_{vc}^{\dagger} f_{\mathrm{Hxc}} P_{vc}, \tag{3}$$

here f_{Hxc} is the kernel of Hartree-exchange–correlation operator

$$\begin{aligned} f_{\mathrm{Hxc}}\left(\mathbf{r},\mathbf{r}'\right) &= f_{\mathrm{H}}\left(\mathbf{r},\mathbf{r}'\right) + f_{\mathrm{xc}}[n]\left(\mathbf{r},\mathbf{r}'\right) \\ &= \frac{1}{|\mathbf{r}-\mathbf{r}'|} + \frac{\delta V_{\mathrm{xc}}[n](\mathbf{r})}{\delta n\left(\mathbf{r}'\right)}, \end{aligned} \tag{4}$$

where $n(\mathbf{r}) = \sum_{i=1}^{N_v} |\psi_i(\mathbf{r})|^2$ encodes the electronic density and f_{xc} is the exchange–correlation potential in LR-TDDFT calculations.

After constructing the Hamiltonian matrix, like other conventional KS-DFT calculations, we need to diagonalize it explicitly to get the excitation wavefunctions X and corresponding excitation energies λ. In our naïve implementation, diagonalization is realized by SYEVD routine in ScaLAPACK [25] with ultra-high complexity of $\mathcal{O}(N_v^3 N_c^3) \sim \mathcal{O}(N_e^6)$. We explain our implementation in Algorithm 1.

We remark that N_r is generally much larger (1000×) than N_e and $N_v \approx N_c \approx N_e$ for a large normalized plane-wave basis set. And we summarize the computational and memory cost of constructing and diagonalizing Hamiltonian in Table 2.

4. Parallel implementation

4.1. Basic design

To fully take advantage of computing resources provided by modern HPC systems, we carefully design a two-level MPI-OpenMP hybrid parallelization strategy along with different forms of data distribution fashions in LR-TDDFT implementation depending on their physical nature.

Our method is written within PWDFT (Plane Wave Density Functional Theory) [26], which forms one separate component of the massively parallel quantum chemistry calculations software package DGDFT (Discontinuous Galerkin Density Functional Theory) [27]. For simplicity in implementation and computational scalability, we apply the local-density approximation (LDA) [28] functional in the KS-DFT and LR-TDDFT calculations.

4.2. Parallel data distribution formula

As we can see from Fig. 1, we design three data distribution schemes for the naïve LR-TDDFT implementation. The first one is column block partition, which means each column of wavefunctions Ψ is distributed to each MPI process according to its column index. This data distribution scheme is approvingly efficient to apply Hartree operator since each MPI task is able to perform fast Fourier transform (FFT) independently in reciprocal space. The second one is row block partition, which means each row of wavefunctions Ψ is distributed to each MPI process based on its column index. This distribution strategy benefits the calculation of the face-splitting product and matrix–matrix multiplication (GEMM). We remark that we use MKL [29] and FFTW [30] to carry out GEMM and FFT operations respectively in our implementation.

We obtain the ground-state wavefunctions from PWDFT, which is stored with the column block partition theme. Then we transform the wavefunctions to row block partition theme to apply the face-splitting product. For the Hamiltonian matrix, we first apply the Hartree operator, which is diagonal in reciprocal space, and then apply the exchange–correlation operator, which is diagonal in real space. To facilitate the calculation steps according to their peculiarities, we use fast Fourier transform (FFT) to convert the P_{vc} from real space to the $\tilde{P}_{vc}$ in reciprocal space ($\{\mathbf{G_i}\}_{i=1}^{N_g}$, N_g indicates grid points in reciprocal space). To apply Fourier transform, the conversion from row block partition to column block partition is carried by MPI_Alltoall routine as demonstrated in Fig. 1(a) and (b). When we finish constructing the Hamiltonian, we need to diagonalize it to obtain excitation energies and corresponding wavefunctions. For the diagonalization step, the two-dimensional block-cyclic partition theme as sketched in Fig. 1(c) is the most advantageous data distribution type to perform direct

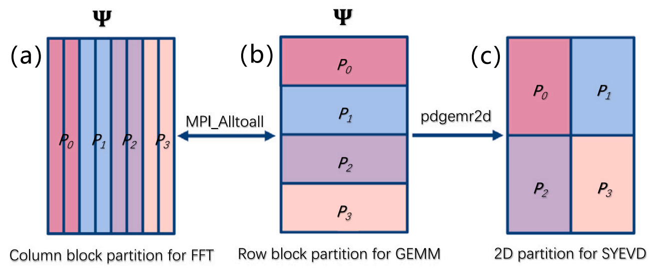

Fig. 1. Parallel data and task distribution schemes of LR-TDDFT. (a) column block partition for FFT, (b) row block partition for GEMM and face face-splitting product (c) 2-D parallelization for diagonalization (SYEVD). The parallel scheme is given with 8 wavefunctions and 4 computing processors as examples.

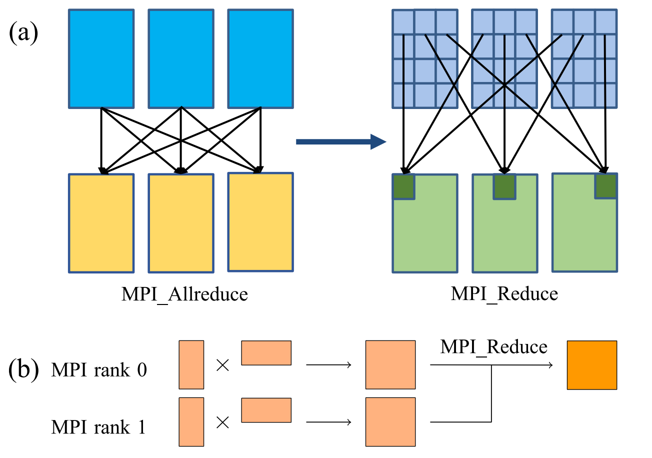

Fig. 2. Data reduction optimization, take the first row of Matrix V_{Hxc} as an example.

diagonalization via the SYEVD routine in the ScaLAPACK library. We perform the data redistribution routine pdgemr2d provided by the ScaLAPACK library to convert the data distribution theme from the row block partition to the two-dimensional block-cyclic partition

4.3. Overlap of computation and communication

As shown in lines 8 and 9 of Algorithm 1, after we perform General Matrix Multiply (GEMM) to get the Hartree-exchange–correlation integrals in every single process,

MPI_Allreduce is used to gather V_{Hxc} in all MPI tasks. There is data dependence because MPI_Allreduce must wait for GEMM to finish computation, in particular when the size of a matrix is large, thus disrupting the overlapping of computation and communication. When the studied system's size increases, although GEMM can be calculated via MKL in a very efficient way, both GEMM and MPI_Allreduce will introduce much time cost. To fully accelerate the process of LR-TDDFT, we make attempts to overlap the step of computation and communication.

First, by analyzing the data partition of LR-TDDFT, we find that in order to calculate the difference between the energy eigenvalues of Kohn–Sham function, not all MPI tasks need to store the entire V_{Hxc} matrix. Therefore, we optimize the data partitioning method shown in Fig. 2. Then we get the result of GEMM, each MPI task only needs to store a part of the V_{Hxc} matrix.

The above attempt brings two benefits. First of all, this new data partitioning method can reduce the memory usage of the MPI process. Second, we do not need to execute MPI_AllReduce to collect the entire V_{Hxc} matrix but use MPI_Reduce to transmit a part of the V_{Hxc} matrix to each MPI task according to the index.

As a result of this attempt, we have eliminated part of the data dependence. In more detail, we can divide the matrix into small pieces

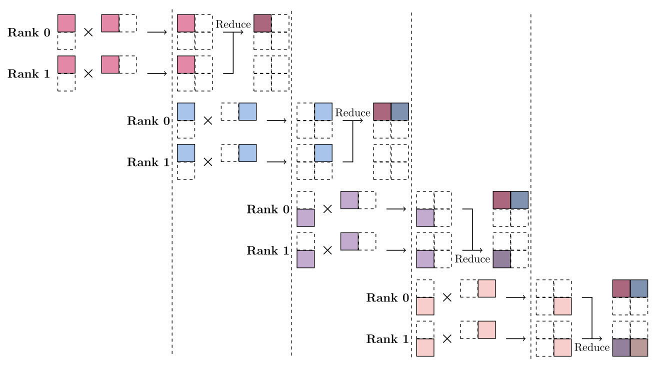

Fig. 3. Pipeline approach of GEMM and MPI_Reduce.

and manually perform GEMM on these small parts. The basic flow of GEMM and reduction is shown in Fig. 3. Once the result of each block is obtained, we can immediately reduce the block matrix to each MPI task through MPI_Reduce. Then, we will allocate the entire V_{Hxc} distributed in each MPI task.

5. Optimizations on numerical algorithm aspect

As we can see from Table 2, the Hamiltonian matrix occupies a large fraction of the memory footprint. For example, when $N_c = N_v = 256$ and double-precision is used during the calculation, each process will hold a matrix of 32 GB, which brings about ultra-high computation cost and communication overhead, hence limiting the studied system size to expand. In this section, we discuss our approach to accelerate LR-TDDFT by implementing numerical algorithms that leverage the inherent properties of low-rank matrices in LR-TDDFT.

5.1. Low-rank approximation in LR-TDDFT by ISDF

As shown in Algorithm 1, all computational operations are inherently based on the two-electron integrals $\left\{\rho_{ij}(\mathbf{r}) := \psi_i(\mathbf{r})\phi_j(\mathbf{r})\right\}_{1\le i\le m, 1\le j\le n}$ (orbital pair product). But when we look into the matrix $P_{vc}(r)$ constructed from valence and conduction orbitals Ψ and Φ, the information beneath it is commonly markedly redundant. In other words, we can use several much smaller matrices to represent it. So exploiting the numerical rank deficiency of the pair products is the cornerstone to reducing the time cost of this operation and all the related computing-intensive operations. Several low-rank tensor approximations have been proposed, including the Resolution-of-the-identity (RI) [31] approximation and the interpolative separable density fitting (ISDF) [32] decomposition. The key spirit of these low-rank approximations is to carefully choose a set of interpolation points N_μ ($N_\mu = cN_r$, where c is a small preconstant) from all the real space grid points N_r in advance, which can give an accurate representation of all orbital-pair products. So we can represent $\psi_i(\mathbf{r})\phi_j(\mathbf{r})$ with the multiplication of two matrices. One matrix can be viewed as the expansion coefficients matrix $C_\mu^{ij}{}_{1\le\mu\le N_\mu}$ (a third-order tensor), whose every single row is extracted from the two-electron integrals matrix according to interpolation points $\hat{r}_\mu$ for $\mu = 1,\ldots,N_\mu$. The other matrix can be viewed as numerical auxiliary basis functions (ABFs) $\left\{\zeta_\mu(\mathbf{r})\right\}_{1\le\mu\le N_\mu}$, for which we will refer to as the interpolating vectors in the rest of the article, so that

$$\psi_i(\mathbf{r})\phi_j(\mathbf{r}) \approx \sum_{\mu=1}^{N_\mu} \zeta_\mu(\mathbf{r}) C_\mu^{ij}. \tag{5}$$

Furthermore, the central idea of the ISDF decomposition different from other low-rank tensor approximations is to decompose the third-order tensor $C_\mu^{ij}{}_{1\le\mu\le N_\mu}$ again, into a transposed block face-splitting

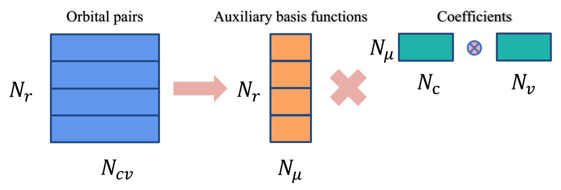

Fig. 4. The ISDF decomposition and its parallel strategy for LR-TDDFT. Each process holds a segment of orbital pairs and auxiliary basis functions, and every process holds the same copy of coefficients.

product of two matrices $C_\mu^{ij} = \psi_i(\hat{\mathbf{r}}_\mu)\phi_j(\hat{\mathbf{r}}_\mu)$. The decomposition scheme of ISDF is shown in Fig. 4.

Thus the Hamiltonian matrix can be rewritten as:

$$H = D + 2C^\dagger(\tilde{V}_{Hxc}C), \tag{6}$$

under ISDF with the auxiliary basis set, the Hartree exchange–correlation integrals V_{Hxc} can be projected in this form:

$$\tilde{V}_{Hxc} = \zeta_\mu^\dagger(f_{\mathrm{Hxc}}\zeta_\mu). \tag{7}$$

The parentheses in Eqs. (6) and (7) represent the order of multiplication.

The time cost of constructing the Hamiltonian (not including ISDF procedure, which we will discuss later) is now significantly reduced to $\mathcal{O}(N_r N_\mu^2 + N_\mu N_v^2 N_c^2)$. Under the ISDF basis set, the first term accounts for the time cost of computing $\tilde{V}_{\mathrm{Hxc}}$ and the second term is the cost of three matrix multiplications and FFTs.

In Section 5.1.1 and 5.1.2, we will discuss the main procedures in the ISDF approximation as a background of our improved method.

5.1.1. *Selecting the interpolation points of ISDF*

Consider a discretized matrix Z of size $N_r \times (N_{cv})$ and find N_μ rows of Z so that the remaining rows of Z can be approximated by the linear combination of the selected N_μ rows. This procedure is so-called interpolative decomposition, and one traditional way is using the randomized sampling QR factorization with column pivoting (QRCP) [33] from real space grid points to attain a low-rank approximation of Z

$$Z^T\Pi = \mathrm{QR}, \tag{8}$$

where Z^T is the transpose matrix of Z. QRCP decomposes Z^T into a product of an $N_{cv} \times N_r$ orthogonal matrix Q and an upper triangular matrix R, and Π is a permutation matrix calculated to ensure the value of the diagonal elements of R form a nonincreasing sequence to facilitate the determination of interpolation points. As we finish the QRCP calculation, the value of the diagonal elements of matrix R indicates how significant the corresponding column of the Z^T matrix is, we choose the largest ones as interpolation points. In order to reduce the cost during QRCP procedure, we set a minimum numerical threshold. When the $(N_\mu + 1)$ th diagonal element of matrix R becomes less than this threshold, the factorization is concluded, and the corresponding grid points are picked as the interpolation points. The leading N_μ columns of the permuted Z^T are considered to be linearly independent. The precision for QRCP to find the suitable interpolation points is promising, however, the matrix Z requires $O(N_r \times N_c \times N_v) \approx O(N_e^3)$ memory and a standard QRCP procedure also cost the computation time of $O(N_e^3)$, which are not quite desirable.

5.1.2. *Computing the interpolation vectors of ISDF*

When the interpolation points are determined and the corresponding coefficient matrix is constructed at the same time, the next step is to compute the interpolation vectors, which form auxiliary basis functions (ABFS). We rewrite Eq. (5) as

$$Z = \Theta C, \tag{9}$$

Eq. (9) is an overdetermined linear system problem with respect to the interpolation vectors $\Theta = \left[\zeta_1, \zeta_2, \ldots, \zeta_{N_\mu}\right]$. In general, we impose the Galerkin condition to solve this overdetermined problem.

$$\Theta = ZC^T\left(CC^T\right)^{-1}. \tag{10}$$

To this end, the solution to Eq. (10) is a least-squares approximation problem of Eq. (9).

In general, ISDF projects the orbital pairs matrix into a much smaller space, which uses N_μ interpolation points to locally express the whole grid points N_r. In our tests, the traditional QRCP procedure for interpolation points chosen, provided by Linear Algebra PACKage (LAPACK) [34], occupies about 90% of the overall ISDF time. So our focus is placed on finding a cheaper method to accurately find the interpolation points.

5.2. *Combining ISDF with K-means clustering*

To further reduce the expensive QRCP procedure in interpolation points selection, we propose a parallel k-Means clustering based interpolation point sampling algorithm in LR-TDDFT. K-Means clustering is one of the most simple yet effective unsupervised machine learning algorithms, which can reveal the underlying correlation of data (electronic correlation effect in this work) by partitioning the real-space grid points into K non-overlapping clusters according to their range of similarity.

In this work, we use the weighted K-means algorithm to determine N_μ non-overlapping clusters from N_r real-space grid points to further choose corresponding interpolation points.

$$\underset{\mathbf{C}_k,\mathbf{c}_k}{\arg\min} \sum_{k=1}^{N_\mu} \sum_{\mathbf{r}_i \in \mathbf{C}_k} w(\mathbf{r}_i)\,|\mathbf{r}_i - \mathbf{c}_k|^2, \tag{11}$$

here, $\mathbf{C}_k$ is the cluster given by

$$\mathbf{C}_k = \left\{\mathbf{r}_i \;\middle|\; |\mathbf{r}_i - \mathbf{c}_k|^2 \le |\mathbf{r}_i - \mathbf{c}_m|^2 \quad \text{for all } i\right\}. \tag{12}$$

The distance between two points in the K-means algorithm is defined as squared Euclidean distances (indicated by $|\mathbf{x} - \mathbf{y}|^2$). Thus, to determine which cluster a grid point belongs to, we need to calculate the mean Euclidean distances $\mathrm{dist}(\mathbf{r}_i, \mathbf{c}_k)$ between this grid point $\mathbf{r}_i$ and all centroids $\mathbf{c}_k$. The centroid of a cluster $\mathbf{c}_k$ is defined as the weighted average of it

$$\mathbf{c}_k = \frac{\sum_{\mathbf{r}_j \in C_k} \mathbf{r}_j w(\mathbf{r}_j)}{\sum_{\mathbf{r}_j \in C_k} w(\mathbf{r}_j)}, \tag{13}$$

and $w(\mathbf{r}_i)$ is the weight function for each real-space grid point. In LR-TDDFT calculations with plane-wave basis set, we define the weight function as Eq. (14) of each row of Z, so it can faithfully represent the norm of orbital pairs

$$w(\mathbf{r}) = \sum_{i=1}^{N_c} \sum_{j=1}^{N_v} |\phi_i(\mathbf{r})|^2 |\phi_j(\mathbf{r})|^2. \tag{14}$$

However, using the original K-Means algorithm with random initialization without concerning any underneath feature of the grid points may yield a terrible convergence problem. Since the grid points of orbital pairs contain specific features, the initialization of centroids should be based on the weight function. At the same time, the weight function vector w(r) is in fact of low rank with plane-wave basis set, which means that we only need to care about the grid points whose weights are non-zero or greater than a threshold during the K-Means procedure. For this reason, we first calculate the weight function at all grid points and remove the points with weights less than the threshold, then initialize N_μ centroids and apply K-Means algorithm only for the

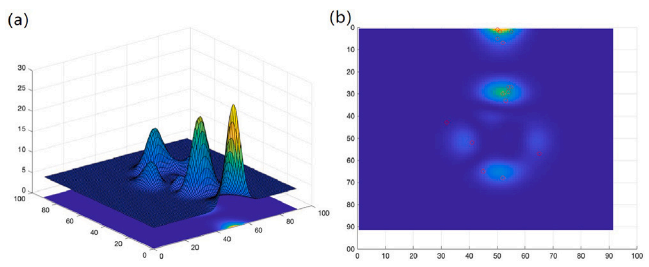

Fig. 5. (a) An example of excitation wavefunctions. (b) Projection of excitation wavefunctions and 15 interpolation points chosen by k-Means clustering (indicated by red dots).

Table 3
Time (in seconds) spent in selecting interpolation procedure of LR-TDDFT calculations.

N_μ	Selecting interpolation points in ISDF	
	QRCP	K-Means
512	10.12	1.61
1,024	42.16	2.85
2,048	147.27	5.57

remaining grid points. Specifically, we choose N_μ grid points as the initial centroids whose weight functions are rather large.

Since the K-Means algorithm can be directly parallelized, its parallel performance is quite satisfying. The classification step is the most time-consuming step and can be locally computed for each group of grid points. After this step, the weighted sum and total weight of all clusters can be reduced from centroids and broadcast to all processors for the next iteration.

To validate the capability of our K-Means approach, according to Table 3, we perform a test of our QRCP and K-Means approach on Si_{64} systems with one processor of Intel Xeon E5-2695 CPU. The results shown in Fig. 5 indicatethat we can get the same interpolation points with a much cheaper time cost. It should be noticed that the improved K-Means approach in LR-TDDFT calculation scales as $\mathcal{O}(N_\mu N_r'^2)$ and N_r' is much smaller than N_r. And the total computational cost for constructing Hamiltonian is $\mathcal{O}(N_r N_\mu^2 + N_\mu N_v^2 N_c^2 + N_\mu N_r'^2)$, which is about 2 orders of magnitude smaller compared to our naïve approach. Also, the memory cost in LR-TDDFT calculations is reduced from $\mathcal{O}(N_r N_v N_c + N_v^2 N_c^2)$ to $\mathcal{O}(N_\mu N_v N_c + N_v^2 N_c^2)$, although real space points N_r is generally larger than $N_v N_c$, the memory cost $\mathcal{O}(N_v^2 N_c^2)$ still consumes an expensive memory footprint, which will be further optimized in Section 5.3.

5.3. Iterative eigensolver for implicitly constructing hamiltonian

Constructing and diagonalizing the Hamiltonian occupy almost half of the total wall clock time. According to Amdahl's law [35], to reach a desired overall speedup performance, we also need to accelerate the Hamiltonian diagonalizing step.

To diagonalize H means solving $HX = \Lambda X$ equation, where X represents the coefficient of excitation wavefunctions (eigenvectors) and Λ presents the excitation energies (eigenvalues). In general, the matrix H is large and we always only need a few eigenvalues and eigenvectors. It means that instead of solving the entire diagonalization problem and then extracting certain eigenvalues, we only need to find a specific eigen-subspace of H with the smallest eigenvalue. To meet this requirement, we use a parallel locally optimal block preconditioned conjugate gradient (LOBPCG) method, which is a conjugate gradient method, to solve the equation in the subspace.

In the LOBPCG method, we use the updating formula:

$$X^{(i+1)} = X^{(i)} * C_1^{(i+1)} + W^{(i)} C_2^{(i+1)} + P^{(i)} C_3^{(i+1)}, \tag{15}$$

where W is the preconditioned gradient constructed from:

$$W^{(i)} = K_i^{-1}(HX^{(i)} - X^{(i)}\Theta^{(i)}). \tag{16}$$

K^{-1} is a precondition to accelerate LOBPCG method:

$$K_i = \epsilon_{i_c} - \epsilon_{i_\mu} - \Theta_I^{(i)}. \tag{17}$$

P is an aggregate direction from the previous step:

$$P^{(i)} = W^{(i-1)} C_2^{(i)} + P^{(i-1)} C_3^{(i)}, \tag{18}$$

and when $i = 1$, we choose $P^{(i)} = 0$. If we mark $S_i = [X^{(i)}, W^{(i)}, P^{(i)}]$, then the key step in the LOBPCG method is to project H onto the subspace S_i ($H \in \mathbb{C}^{m\times m}$, $S_i \in \mathbb{C}^{m\times 3k}$, $H_s = S_i^\dagger H S_i \in \mathbb{C}^{3k\times 3k}$ and $C^{(i)} = [C_1^{(i)}, C_2^{(i)}, C_3^{(i)}]^T$) and solve the projected eigenvalue problem $H_s C^{(i+1)} = S_i^\dagger \Lambda_i S_i C^{(i+1)}$. When the subspace S and coefficients C reach a convergence, the corresponding excitation wavefunctions $X = SC$ can be directly computed.

For each iteration in the LOBPCG method, the total computation cost is $3kN_c^2 N_v^2 + (3k)^2 N_{cv} + (3k)^3 \sim kO(N_e^4)$.

Although LOBPCG is a standard procedure in iterative diagonalization, the explicit Hamiltonian cost a $O(N_e^4)$ memory footprint. We notice that $H_s = S_i^\dagger H S_i$ can be expanded as $H_s = S_i^\dagger D S_i + 2S_i^\dagger\{P_{vc}^\dagger[(v_{\mathrm{H}} + f_{\mathrm{xc}})(P_{vc}S_i)]\}$, which means combining with ISDF decomposition, the H can always keep a factored form. After changing the order of calculations, the total computational cost of implicitly constructing and diagonalizing the Hamiltonian H_s is $3kN_\mu N_c N_v + 3kN_\mu N_\mu + (3k)^2 N_\mu + (3k)^3 \sim kO(N_e^3)$. The pseudocode of the implicit LOBPCG method is demonstrated in Algorithm 2.

Algorithm 2 Implicit LOBPCG method for solving the LR-TDDFT eigenvalue problem $Hx_i = \lambda_i x_i, i = 1, 2, \ldots, k$.

```
Input:   Hamiltonian $H$ and initial wavefunctions $\{x_i\}_{i=1}^k$.
  Initialize the trial subspace $S_1 = [X^{(1)}, W^{(1)}]$ and orthonormalize $S_1$.

  while convergence not reached do
    Project $H$ onto the subspace $S_i$: $H_s = S_i^\dagger D S_i + 2S_i^\dagger\{P_{vc}^\dagger[(v_{\mathrm{H}} + f_{\mathrm{xc}})(P_{vc}S_i)]\}$;
    Solve the projected eigenvalue problem $H_s C^{(i)} = C^{(i)}\Theta_i$ and obtain the coefficients $C = [C_1^{(i)}, C_2^{(i)}, C_3^{(i)}]^T$ and eigenvalues $\Theta_i$;
    Compute $X^{(i)} \leftarrow S_i C^{(i)}$, preconditioned gradient vectors $W^{(i)} = K^{-1}(HX_i - X_i\Theta_i)$ and aggregate direction $P^{(i)} = W^{(i-1)}C_2^{(i)} + P^{(i-1)}C_3^{(i)}$;
    Construct the subspace $S_{i+1} \leftarrow [X^{(i)}, W^{(i)}, P^{(i)}]$;
  end while
  Update $\{x_i\}_{i=1}^k \leftarrow X^{(i)}$.
Output:
  Eigenvalues $\{\lambda_i\}_{i=1}^k$ and wavefunctions $\{x_i\}_{i=1}^k$.
```

The complexity after each step is summarized in Table 4. As we can see, Implicit-Kmeans-ISDF-LOBPCG version ((5) in Table 4) significantly reduces the computation and memory cost by nearly 2 orders of magnitude.

6. Optimizations on new sunway heterogeneous architecture aspect

Although ISDF-LOBPCG method can provide significant performance improvement, it may lead to a slight degradation in accuracy, which we will discuss in detail in Section 7. In certain scenarios, ensuring accuracy is of the utmost importance even if it means sacrificing simulation performance. In this section, we showcase an alternative method that harnesses the immense computing power of the new Sunway supercomputer to accelerate LR-TDDFT without accuracy loss.

Table 4
Computational and memory complexity for five different versions for constructing and diagonalizing Hamiltonian in the excited-state electronic structure calculations, including the naïve case and four-level optimized cases. Notice that $N_r \approx 1000 \times N_e$, $N_\mu \approx 10 \times N_e$, $N_v \approx N_c \approx N_e$ and $1 \le k \ll N_e$ in the plane-wave basis sets.

LR-TDDFT versions		Constructing Hamiltonian		Diagonalizing Hamiltonian	
		Computation	Memory	Computation	Memory
(1)	Naïve	$\mathcal{O}(N_v^2 N_c^2 N_r + N_v N_c N_r)$	$\mathcal{O}(N_v^2 N_c^2 + N_r N_v N_c)$	$\mathcal{O}(N_r^2 N_v^2 N_c^2)$	$\mathcal{O}(N_v^2 N_c^2)$
(2)	QRCP-ISDF	$\mathcal{O}(N_r N_\mu^2 + N_\mu N_v^2 N_c^2 + N_\mu N_r^2)$	$\mathcal{O}(N_v^2 N_c^2 + N_\mu N_v N_c)$	$\mathcal{O}(N_r^2 N_v^2 N_c^2)$	$\mathcal{O}(N_v^2 N_c^2)$
(3)	Kmeans-ISDF	$\mathcal{O}(N_r N_\mu^2 + N_\mu N_v^2 N_c^2 + N_\mu N_r'^2)$	$\mathcal{O}(N_v^2 N_c^2 + N_\mu N_v N_c)$	$\mathcal{O}(N_r^2 N_v^2 N_c^2)$	$\mathcal{O}(N_v^2 N_c^2)$
(4)	Kmeans-ISDF-LOBPCG	$\mathcal{O}(N_r N_\mu^2 + N_\mu N_v^2 N_c^2 + N_\mu N_r'^2)$	$\mathcal{O}(N_v^2 N_c^2 + N_\mu N_v N_c)$	$k\mathcal{O}(N_v^2 N_c^2)$	$\mathcal{O}(N_v^2 N_c^2)$
(5)	Implicit-Kmeans-ISDF-LOBPCG	$\mathcal{O}(N_r N_\mu^2 + N_\mu N_v N_c + N_\mu N_r'^2)$	$\mathcal{O}(N_v^2 N_c^2 + N_\mu N_v N_c)$	$k\mathcal{O}(N_\mu N_v N_c)$	$\mathcal{O}(N_\mu^2)$

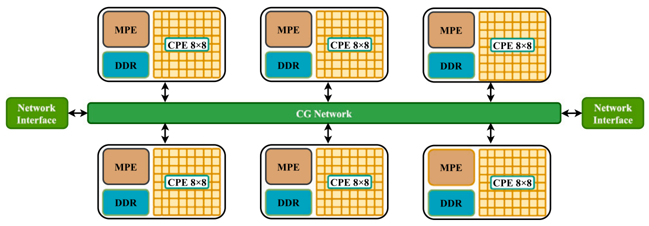

Fig. 6. SW26010Pro many-core processor on the new Sunway supercomputer.

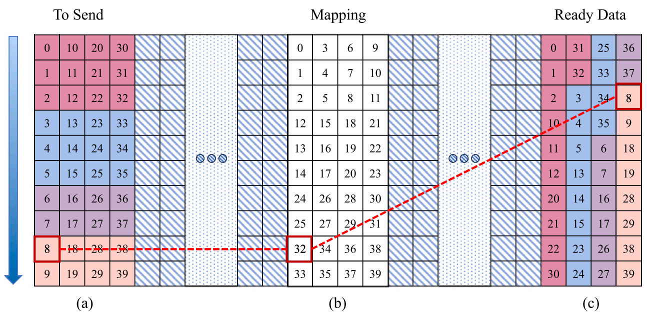

Fig. 7. The memory layout prior to task distribution scheme transformation, where (a) displays the column partition data, and each element is numbered according to its position in memory, (b) represents the matrix which maps corresponding element in (a) to its target position in (c), and (c) represents data ready for communication in which elements for the same process are continuous. The memory addresses are contiguous along the column direction of the matrix.

6.1. *Architecture of the new sunway supercomputer*

The new generation of Sunway supercomputer incorporates the latest SW26010Pro many-core processor as shown in Fig. 6. The processor consists of six core groups (CGs), each containing a management processing element (MPE) and 64 computing processing elements (CPEs) equipped with 16 GB DDR4 memory.

Although the new Sunway supercomputer is able to unify the individual memory of each core group into a consolidated 96 GB memory, it is important to note that certain essential numerical libraries such as FFTW [30], BLAS [36], and LAPACK [34] have not yet been adapted to accommodate this particular scheme. As a result, DFT calculations must be conducted using a more conventional scheme, where each core group is considered an independent processor with its own dedicated 16 GB memory.

6.2. *Cache-friendly on-the-fly matrix mapping*

In this subsection, we present our inventive methodologies of improving the granularity of memory access to take advantage of the bandwidth and confining it to a smaller range of addresses to make it cache-friendly. These approaches lead to notable enhancements in memory access while simultaneously reducing memory consumption to a significant extent.

6.2.1. *Background description*

As discussed in Section 4.2, DFT calculations entail a transformation in the data and task distribution scheme, leading to numerous and frequent memory accesses to make distributed data ready for communication.

Moreover, given the considerable size of wavefunctions Ψ, it is impractical to accommodate the distributed data with the last level cache. That is to say, the majority of elements are unable to effectively exploit the cache hierarchy, resulting in poor performance, especially on the new Sunway. These findings are substantiated by the experimental results outlined in Table 5.

Processors such as the Intel Xeon E5-2695, comprising multiple cores, typically allocate one process per physical core, resulting in the sharing of available memory bandwidth among processes within the same processor. Conversely, in DFT calculations, each core group of SW26010Pro is assigned one process, which means it can fully exploit the entirety of the memory bandwidth.

Therefore, we can use large blocks of sequential accesses in memory to reduce the performance degradation caused by higher access latency, and confine memory access to physically closer memory address to improve cache behavior.

6.2.2. *Design and implementation*

To illustrate task distribution transform, Fig. 7 presents a simplified memory layout prior to the distribution scheme changes from the column block partition to the row block partition, in which four processes are involved, and elements destined for the same process are color-coded accordingly.

We present a common case that the rows of the matrix cannot be exactly divided by the number of processes. This means that certain blocks may contain one additional element compared to others. And Fig. 7 visually depicts this situation.

As depicted in the figure, some elements keep their relative position unchanged during the data transformation. This inspires us to combine such kinds of elements into much larger blocks, which will be considered a single unit in transformation. It is crucial to emphasize that in actual cases, these unified blocks encompass a substantial number of elements, ranging from hundreds to thousands, depending on the number of processes. This characteristic greatly contributes to the efficiency of our implementation.

At the outset, we partition the original matrix into two sections as shown in state 1 and state 2 in Fig. 8(a), where blocks within the same section possess an equal number of elements. The upper section's blocks have one additional element than those in the lower section. The boundary between these sections is illustrated as a dashed line. It is important to note that the blocks within each section are not contiguous in memory. This partitioning allows us to differentiate blocks with varying numbers of elements. Meanwhile, we also introduce a small array in Fig. 8(b), where the ith element within this array refers to the ith block of the original matrix in memory order, while the stored value indicates the desired order in the final state.

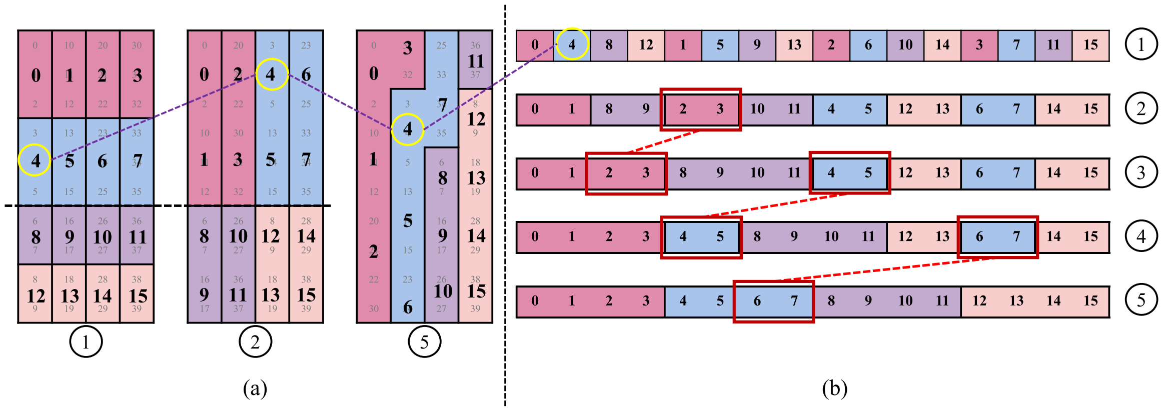

Fig. 8. The procedures of packing elements into blocks and conducting the in-place data movement. In (a), the detailed contents of the original matrix are provided for states 1, 2, and 3, showcasing the ordering of each element in the column data partition using small grey numbers, along with the corresponding block indicated by **larger bold numbers**. In (b), the contents of the blocks array are displayed for each state. Each rectangle with a specific number corresponds to the block with the same number in (a). The number assigned to each block signifies its target order among all the blocks. Blocks intended for the same process are painted with the same color.

Table 5
The execution time (in seconds) of the mentioned procedures, both before and after packing, using 1000 atoms with different core groups.

Atoms	CGs	Time	Time (Optimized)	Speedup
1000	50	43.08	22.859	1.88
1000	100	21.332	4.02	5.31
1000	125	16.993	1.751	9.70
1000	200	11.04	1.866	5.92
1000	250	8.879	1.189	7.47
1000	500	4.587	0.439	10.45
1000	1000	2.512	0.219	11.47

Within each section, we proceed with block exchanges to rearrange the position of each block. This exchange is facilitated by two small mapping matrices, with each matrix corresponding to one section. Each item within the mapping matrices corresponds to a specific block and indicates the desired order of the corresponding block within the procedure. Given the deterministic transformation pattern, obtaining such a mapping matrix is not a challenging task. Consequently, the matrix transitions to state 2.

Through this step, some blocks have already attained their correct order in relation to their adjacent blocks in the final outcome. Therefore, in subsequent operations, these blocks can be treated as larger blocks, enhancing the granularity of data movement. For instance, blocks '2' and '3' will move ahead of blocks containing '8' and '9', leading the matrix to state 3. Following this, we shift blocks containing '4' and '5' forward, reaching state 4. Further moving blocks with '6' and '7' forward completes the matrix's transition to the final target state. It is evident that the number of movements required is precisely equal to the number of columns minus one.

It is worth highlighting that when transitioning from the row partition to the column partition, a similar implementation is needed. Therefore, we will treat these transformations as a unified entity when evaluating the efficacy of our optimization.

6.2.3. Results and analysis

To assess the efficiency of this optimization, we conduct simulations of Si_{1000} with varying numbers of core groups, with each core group assigned one process. The experimental results are summarized in Table 5. The findings demonstrate that the optimization of packing can yield substantial reductions in the costs associated with random memory access, resulting in a remarkable speedup of up to 11.47x. The significant improvement in performance underscores the effectiveness of this optimization technique.

The results in Table 5 demonstrate a direct relationship between the number of CGs and the execution time in the original implementation. To be specifically, the execution time of the original version is roughly inversely proportional to the number of CGs. In other words, it is approximately proportional to the data volume, as the fixed volume of data is distributed across different processes. A reasonable explanation for this is that the original implementation can barely utilize the cache hierarchy; nearly all data accesses must reach the physical memory, which explains why the execution time correlates with the data volume. In contrast, with the increase of the number of processes, the optimized implementation is more likely to access the CPU cache rather than the physical memory, leading to a tendency for increased speedup. As an example, due to the improved cache behavior, a substantial speedup of 5.68x is observed in our optimized implementation when comparing the execution time of 100 core groups (4.02 s) to that of 50 core groups (22.859 s). These compelling outcomes clearly demonstrate the efficacy of our optimizations in significantly improving the cache hierarchy hit rate. Additionally, our optimization brings significant relief to memory pressure by eliminating the need for a mapping matrix and a buffer of equivalent size to the original data.

6.3. Acceleration by CPEs

The predominant floating performance on the new Sunway supercomputer is provided by plentiful CPEs. Therefore, it is of great interest to accelerate the calculation by adopting CPEs.

On the new Sunway, CPEs do not have direct access to the main memory of the core group. Instead, each CPE is equipped with its own local device memory (LDM) of 256 KB. The LDM can be configured in different modes to suit specific requirements. One configuration option is to set LDM as a cache, which operates transparently to the programmer and program. Alternatively, it can be configured as a distributed shared space, necessitating explicit data loading from the main memory through direct memory access (DMA). Additionally, a hybrid mode is available where a portion of the LDM functions as a cache while the remaining portion serves as shared space.

Several essential numerical libraries, such as ScaLAPACK, LAPACK, and swFFT (a library for fast Fourier transformation on the new Sunway), have already been integrated into the new Sunway system. With the availability of these transported and fully optimized libraries, we can employ them to carry out various computations such as general matrix multiplication (GEMM), and fast Fourier transformation (FFT). These libraries empower us to efficiently perform these operations, leveraging the computational capabilities of the CPEs and exploiting the optimized functionality of the Sunway system.

Furthermore, recent research introduces a unified programming framework called ShenWei Universal C/C++ (SWUC) for deploying computational workloads onto CPEs [37]. This framework provides a

Algorithm 3 An example of calculating $output[i] = output[i] * input[i]$, $i = 1, 2, \dots, gridNum$ using CPEs, where CRTS_tid is the id of a CPE.

```
1:  Function {SLAVE} {&STEP_SIZE, &gridNum,
2:                    &output, &input}
3:  /* Allocated space in LDM of each CPE. */
4:  DataType output_cpe[STEP_SIZE]
5:  DataType input_cpe[STEP_SIZE]
6:  /* Divide the entire computing task into STEP_SIZE division to fit in the
    size of LDM. */
7:  for i=CRTS_tid*STEP_SIZE; i<gridNum; i+=STEP_SIZE do
8:    isz ← min {gridNum-i, STEP_SIZE}
9:    /* Load data of STEP_SIZE into LDM. */
10:   CRTS_dma_get(output_cpe,
11:       &output[i], isz*sizeof(DataType))
12:   CRTS_dma_get(input_cpe,
13:       &intput[i], isz*sizeof(DataType))
14:   for ii ∈(0, isz) do
15:     output_cpe[ii] *= input_cpe[ii]
16:   end for
17:   /* Write data of STEP_SIZE into main memory. */
18:   CRTS_dma_put(&output[i],
19:       output_cpe, isz*sizeof(DataType))
20: end for
21: EndFunction
22: call SW_RUN( SLAVE )
```

solution to accelerate calculations that may not benefit from existing numerical libraries. Thanks to several new attributes and compiler directives, writing codes running on CPEs and MPE can be achieved by calling for lambda expressions, in which we divide data for each CPE and specify the codes to be executed.

To identify performance bottlenecks, we analyze the runtime information of function calls or computing procedures and deploy the workload to CPEs. Algorithm 3 demonstrates an example of deploying a multiplication series to CPEs. In this algorithm, a function named SLAVE, which is executed on each CPE, is defined and passed as a parameter to SW_RUN, which is an interface to utilize the mentioned framework. To accommodate the limited size of LDM, we divide the entire computing task into smaller divisions, each of size STEP_SIZE. This ensures that the temporary variables output_cpe and input_cpe fit within the available capacity of the LDM. We use the interface CRTS_dma_get to load data from the main memory to LDM and CRTS_dma_put to write results back to the main memory.

Given that this particular optimization significantly contributes to the overall acceleration effect, we will not discuss the experimental results of this aspect separately. Instead, we will analyze and evaluate it as part of the overall optimization results in Section 7.

6.4. Scalability enhancement

The solution of eigenproblems using SYEVD in ScaLAPACK is a computationally intensive task that demands significant time. More unfortunately, the scalability of SYEVD on the new Sunway suffers from terrible behavior due to suboptimal underlying implementation. As a result, this leads to increased communication frequency and an imbalanced workload distribution across processors. The scalability issue becomes particularly prominent when a large number of core groups are utilized, further amplifying the execution time required.

To alleviate this situation, we have implemented a dynamic approach that selects the optimal number of processes for performing SYEVD. On the Sunway platform, the execution time of SYEVD initially decreases as the number of core groups increases, reaching its peak performance at a certain number denoted as P_t. However, beyond this point, the execution time rapidly increases. Therefore, we dynamically

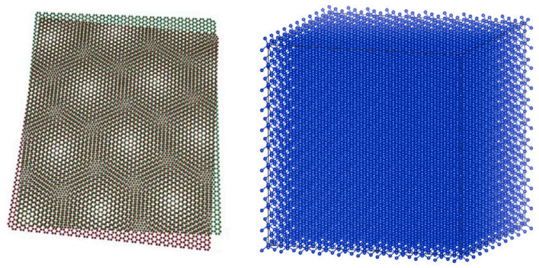
(a) (b)

Fig. 9. Atomic configurations of (a) MATBG and (b) bulk silicon with 4096 atoms.

Table 6
The execution time in seconds of SYEVD with the simulation of silicon systems containing 1728 atoms.

Number of CGs	108	216	432	864	1728
Naive	56.3	71.6	174.1	172.1	161.7
Optimized	56.3	56.3	56.3	56.3	56.3
Speedup	1	1.27	3.09	3.06	2.87

choose the most suitable number of core groups as the minimum value of the allocated core groups and P_t. Our experimental results in Table 6 demonstrate that this approach effectively mitigates performance degradation.

7. Numerical results and analysis

7.1. Setup of the test physical systems and testing environment

The testing systems shown in Fig. 9 include two parts: (1) cubic silicon systems and (2) Magic angle twisted bilayer graphene with 1180 atoms. For cubic silicon systems, we use various choices of crystal silicon systems with 64, 216, 512, 1000, 1728, 2744, and 4096 silicon atoms labeled by Si_{64}, Si_{216}, Si_{512}, Si_{1000}, Si_{1728}, Si_{2744}, and Si_{4096}, respectively.

We apply the Hartwigsen Goedecker Hutter (HGH) norm-conserving pseudopotential in all of the following tests. The total number of real-space grid points N_r is determined by the kinetic energy cutoff (E_{cut}) defined as $\left(N_r\right)_i = \sqrt{2E_{\text{cut}}}L_i/\pi$, where L_i is the length of each supercell along each (x, y and z) coordinate direction. Without additional illustrations, the kinetic energy cutoff in our experiments is 20 Hartree. For example, the number of real-space grid points for a wavefunction matrix in Si_{4096} is $N_r = 166 \times 166 \times 166 = 4{,}574{,}296$.

In our experiments, we conducted tests of numerical algorithms on the National Energy Research Scientific Computing Center (NERSC)'s Cori supercomputer with parameters $nc = 128$, $nv = 128$, and $Extra\_state = 256$. These parameters have a significant impact on the execution of the diagnosis matrix. On Sunway, due to the absence of low-rank algorithms, we reduce these parameters in order to obtain comparable execution times, which are $nc = 64$, $nv = 64$, and $Extra\_state = 64$.

For experiments on NERSC's Cori supercomputer, we run our code on the Haswell partition, whose each node has two sockets, each socket is populated with a 2.3 GHz 16-core Haswell processor (Intel Xeon Processor E5-2698 v3) and 128 GB DDR4 2133 MHz memory. Each core supports 2 hyper-threads and has two 256-bit-wide vector units. Each core has a theoretical peak performance of 36.8 Gflops double-precision operations. If not mentioned particularly, we apply 8 MPI tasks per computing node (4 OpenMP threads per MPI process) on NERSC's Cori supercomputer.

For experiments on the new Sunway, we apply each MPI task per core group.

Table 7
The three lowest excitation energies and corresponding relative errors of H_2O (E_{cut} = 100.0 Ha, N_v = 20 and N_c = 4) system and Si_{64} system (E_{cut} = 50.0 Ha, N_v = 128 and N_c = 50).

QE	LR-TDDFT	LOBPCG-ISDF	$\Delta\ E_1$	$\Delta\ E_1$
		Single water molecule H_2O		
0.398312	0.397830	0.397829	0.121%	0.121%
0.550416	0.546664	0.546664	0.682%	0.682%
0.729568	0.732786	0.732785	−0.441%	−0.441%
		Periodic bulk silicon Si_{64}		
0.044350	0.043942	0.0439429	0.920%	0.918%
0.044350	0.043942	0.0439429	0.920%	0.918%
0.044350	0.043942	0.0439429	0.920%	0.918%

7.2. Numerical accuracy

7.2.1. Numerical algorithm acceleration

We compare our naïve version (LR-TDDFT) code and Implicit-Kmeans-ISDF-LOBPCG version (ISDF-LOBPCG) with the current state of the art software Quantum Espresso (QE) [38], which serves as an accuracy benchmark. Due to the bad scalability of QE, we use H_2O and Si_{64} as our benchmark parts. We choose the system of one H_2O molecule with the simulation boxes $11.000 \times 11.000 \times 11.000$ Å^3 and the system of 64 silicon atoms (Si_{64}) with the simulation boxes $20.525 \times 20.525 \times 20.525$Å^3. The calculations are performed using the Casida calculations and the relative excited energy errors are defined by:

$$\Delta E_1 = (E_{\mathrm{QE}} - E_{\mathrm{LR\text{-}TDDFT}})/E_{\mathrm{QE}}$$
$$\Delta E_2 = (E_{\mathrm{QE}} - E_{\mathrm{ISDF\text{-}LOBPCG}})/E_{\mathrm{QE}} \tag{19}$$

Table 7 lists the results. We find a good agreement between the results of QE and LR-TDDFT, with a small difference in excitation energies for an identical ordering of states. Our optimizations will only introduce little error, as small as 0.001% in relative error, which is fairly negligible. Results mean that we already dismiss almost all redundant computational costs beneath LR-TDDFT calculations. The accuracy reaches the level we need and results that the accuracy will further improve as kinetic energy cutoff increases.

7.2.2. Heterogeneous architecture acceleration

As for our implementation on the new Sunway supercomputer, we adopt the optimization that corresponds to heterogeneous architecture without any numerical methods, which means, this implementation is exactly as accurate as the naive version.

7.3. Algorithm convergence

The algorithms proposed in this work do not affect the systematic convergence of LR-TDDFT, as demonstrated in many previous studies. For example, our prior work successfully applied hybrid-LR-TDDFT to analyze two-dimensional MoS2 consisting of 216 atoms, confirming the capability of the naive implementation to conduct physical research [39]. Furthermore, numerous studies have documented the effectiveness of both the ISDF [40] and LOBPCG [14] methods, each confirming their independent convergence. Furthermore, to demonstrate the convergence of our method unequivocally, we present the convergence curves for the total energy of the ground state of a silicon system containing 1000 atoms below. As illustrated in Fig. 10, the total energy of the ground state of the system reach a convergence within 15 iterations.

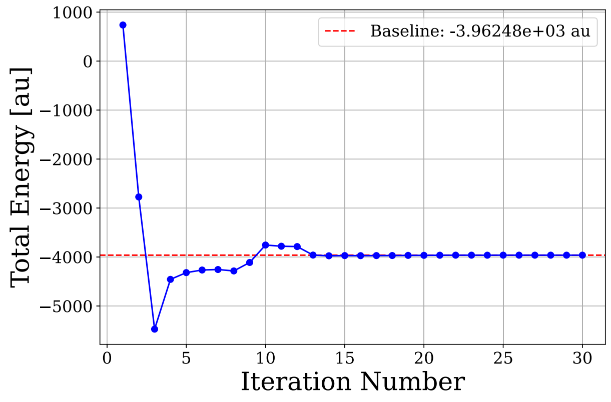

Fig. 10. The convergence curve for the total energy of silicon system with 1000 atoms.

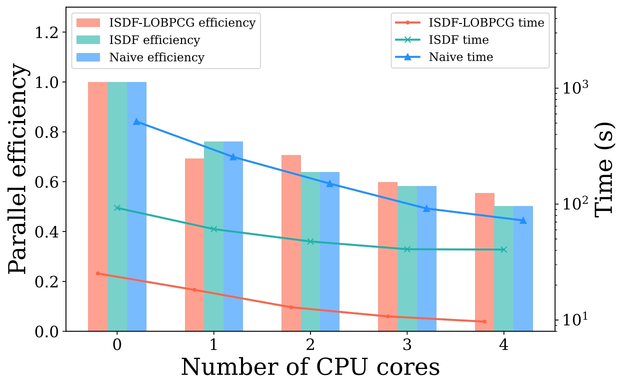

Fig. 11. Strong scaling: the wall clock time and parallel efficiency with respect to the number of CPU cores. Lines denote time (in seconds) and bar denotes parallel efficiency.

7.4. Strong scaling

7.4.1. Numerical algorithm acceleration

Firstly, we will discuss strong scaling on NERSC's Cori supercomputer, which is mainly based on numerical algorithms. As shown in Fig. 11, we present the strong scaling performance of 3 versions of our code: Naïve version, ISDF version and ISDF-LOBPCG version (corresponding to (1), (3) and (5) in Table 4). The testing system contains 1000 silicon atoms and the real space points $N_r = 104 \times 104 \times 104 = 1{,}124{,}864$. We evaluate strong scalability by parallel efficiency defined in Eq. (20), and the speedup is compared with wall clock time in 128 CPU cores.

$$\text{Parallel Efficiency} = \frac{\text{Speedup}}{\text{Multiple of CPU cores}} \tag{20}$$

The parallel efficiency of our naïve design maintains above 50% when scaling to 2048 processing cores. This result is quite acceptable among LR-TDDFT calculations with plane-wave basis set since we need to do a series of collective communication in the global domain.

Also, we give a more detailed analysis by splitting the wall clock time during the procedure of constructing the Hamiltonian into 4 parts: (1) K-Means, (2) FFT, (3) MPI, and (4) GEMM and Allreduce. As Fig. 12 shows, due to our parallel design, the K-Means, GEMM, FFT, and even MPI procedure maintain a very convincing strong scaling performance till 2048 CPU cores. But to implement the implicit method, we transform the Hartree-exchange–correlation integrals from a single GEMM operation to a serial of GEMMs and an MPI_Allreduce, which hinders the total time speedup from ideal. Since MPI collective communication routines will bring in extra overhead. To maintain good speedup and scale the system to a larger size, the implicit Hamiltonian method is

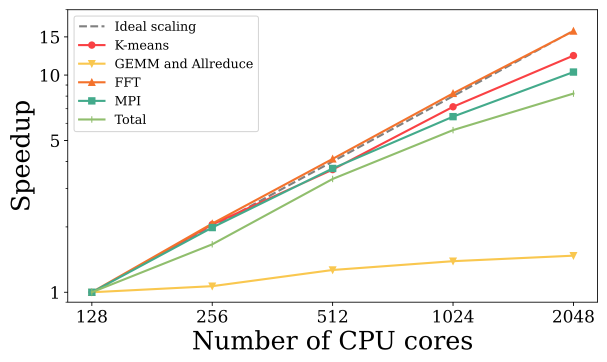

Fig. 12. Strong scaling performance of constructing Hamiltonian step in the LR-TDDFT calculations.

indispensable, so this method is a trade-off between efficiency and strong scaling. But in our test, GEMM and Allreduce step will only cost 12.87% of the total time of constructing Hamiltonian, the small sacrifice of strong scaling is quite worthy.

In particular, we test Si_{4096} system with 8192 and 12,288 processing cores and bind 16 OpenMP threads with each MPI process. The corresponding wall-clock time is 14.02 and 10.70 s, with a strong scaling performance of 87.34%. Since increasing the number of OpenMP threads can reduce the processes within the calculation, it can straightforwardly reduce the communicational cost, hence improving strong scalability when we apply a large number of CPU cores. This unprecedented speed enables the electronic structure exploration of large-scale systems containing more than 4000 atoms by performing LR-TDDFT with a very low computation cost.

7.4.2. Heterogeneous architecture acceleration

Regarding our implementation on the new Sunway, we present detailed results in Fig. 13. Unlike previous results, the dominant procedures on the Sunway platform include memory access, FFT, GEMM, SYEVD, and other calculations. From Fig. 13, we can see that memory access contributes to more than one-third of the total execution time when using 50 core groups (CGs). However, this contribution is significantly reduced when using 100 CGs. Furthermore, when employing more than 250 CGs, the impact of memory access becomes negligible and its execution time is even lower than that of MPI communication. The FFT and GEMM procedures exhibit good scalability as the number of CGs increases. Similarly, MPI communication demonstrates favorable scalability when utilizing less than 500 CGs. As for SYEVD, as discussed earlier, we have implemented a cutoff for the number of processors to maintain reasonable scalability, which explains why its execution time remains unchanged when using more than 100 CGs. The original result of SYEVD is even worse.

7.5. Weak scaling

7.5.1. Numerical algorithm acceleration

For implementation on the Cori supercomputer, in particular, our method can significantly reduce the memory cost during calculation steps of LR-TDDFT simulation, so we can use far fewer computing resources to study a much larger physical system. We use LR-TDDFT-optimized code to test Si_{512}, Si_{1000}, Si_{1728}, Si_{2744} and Si_{4096} systems with 1024 cores and we bind single core to a process, corresponding time is 3.58, 10.23, 26.95, 35.58 and 41.89 s. This result suits our computational complexity well.

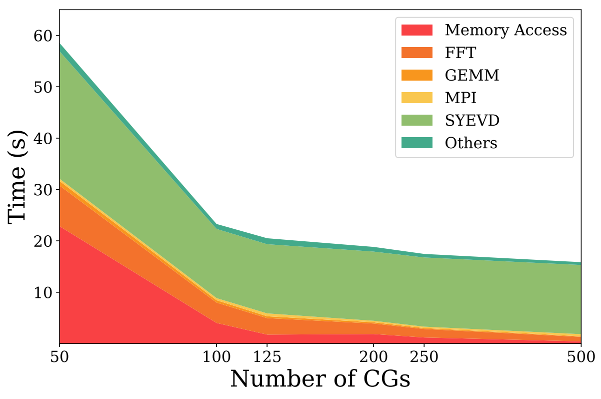

Fig. 13. Strong scaling performance on the new Sunway supercomputer.

Table 8
The wall-clock time (in seconds) and speedups of different sizes of 3D bulk silicon systems.

Systems	Naïve	ISDF-LOBPCG	Speedup
Si_{64}	3.19	0.24	13.06
Si_{216}	6.95	0.70	9.89
Si_{512}	14.74	1.89	7.79
Si_{1000}	32.15	5.13	6.26

7.5.2. Heterogeneous architecture acceleration

For our implementation on the new Sunway, we are unable to perform tests on Si_{4096} due to memory limitations when the Ecut is set to 20. As for Si_{512}, Si_{1000}, Si_{1728}, Si_{2744}, the time to solution is 12.46, 15.35, 32.40 and 45.74 s accordingly with 512 CGs. Considering the much more communication costs of the new Sunway, the result also suits the computational complexity.

7.6. Speedup

7.6.1. Numerical algorithm acceleration

For our implementation on the Cori supercomputer, we further reduce the computation resources and bind a single core to an MPI process, thus each process holds only 4 GB of memory. We evaluate tests on different sizes of systems with Naïve and ISDF-LOBPCG version ((1) and (5) in Table 4) code. We observe an average speedup of 9.254x, which is quite convincing. And as sketched in Fig. 11, when we apply larger computation resources, we also observe an average of 12.58x speedup. In fact, among all of our numerical results, the average speedup under our optimizations is over 10x, as you can see from Table 8. Combining the accuracy property, our method can reach quite faster calculations with fewer resources.

7.6.2. Heterogeneous architecture acceleration

In our implementation on the new Sunway platform, we conducted experiments to compare the performance of our optimized version with the naive version that was ported onto the new Sunway. The results, presented in Fig. 14, demonstrate a trend where the speedup ratio decreases as the number of core groups (CGs) increases. This phenomenon can be attributed to the decrease in computing workload that occurs as the number of CGs increases.

Our optimization achieves the highest speedup of 80.5 when the number of CGs is set to 100. This remarkable improvement can be attributed to the results presented in Table 5, where time for memory access decreases dramatically when involved CGs change from 50 to 100.

Furthermore, when utilizing 500 core groups, our optimization achieves a remarkable speedup of 23x. This emphasizes the effectiveness of our approach in improving the computational performance on the new Sunway platform.

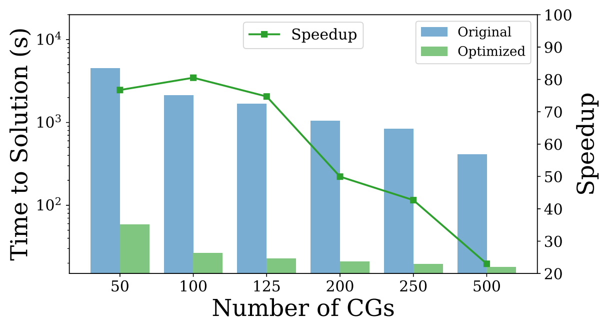

Fig. 14. Speedup on the new Sunway, compared with the naive version.

8. Discussion

The SW26010Pro processor exhibits notable differences compared to the Intel Xeon processor, including clock frequency, cache hierarchy, and the architecture of the many-core design. In certain procedures that are well-suited for the CPE architecture, such as FFT and GEMM, which can take advantage of plentiful CPEs, a core group performs better on the SW26010Pro processor compared to a single core on the Intel Xeon processor. However, for workloads that are not suitable for CPEs, such as communication and SYEVD, a core group can be weaker.

To compare the two optimization schemes, the numerical algorithms demonstrate a significant acceleration of up to ten times by introducing a controllable numerical error as a trade-off. Additionally, the achievement of reducing memory usage by an order of magnitude has led to a substantial reduction in memory consumption. This optimization scheme exhibits good strong and weak scalability, performing well in terms of both increasing computational workload and decreasing workload.

On Sunway, we apply optimizations aimed at full utilization of hardware architecture. As a result, our optimization achieves an impressive speedup of 80.5 compared to the original version with no accuracy loss. However, it is important to acknowledge that the acceleration achieved by acceleration units tends to diminish with the increase of computing hardware, which means that it can hardly maintain perfect scalability. Additionally, applications on heterogeneous platforms heavily depend on the underlying numerical libraries.

9. Conclusion

In this work, we employ two distinct optimization schemes to accelerate linear-response time-dependent density functional theory (LR-TDDFT) calculations. One scheme focuses on the Intel platform, primarily leveraging numerical algorithms with acceptable and controllable accuracy loss. The other scheme targets the Sunway platform, utilizing heterogeneous architecture to achieve impressive speedup. We compare and analyze these two schemes and conduct substantial experiments to evaluate them.

The optimization scheme based on the numerical algorithms achieves a speedup of over 10x in LR-TDDFT calculations while maintaining admirable accuracy. Furthermore, by testing the Si_{4096} system on 12,288 cores, we demonstrate good strong scalability even with a large number of CPU cores. These advancements enable us to push the boundaries of first-principles excited-state simulations. Meanwhile, the acceleration based on heterogeneous architecture achieves an impressive speedup of up to 80x. However, due to scalability issues and memory limitations, it is only feasible for simulations involving up to 2744 atoms. In its current state, this acceleration scheme is more suitable for systems with 1k to 3k atoms. Nonetheless, these achievements provide a significant advancement in the realm of first-principles excited-state simulations. At the method level, we also acknowledge the significant achievements of the novel Sternheimer method in LR-TDDFT and other areas [41–43]. However, we found that compared to the Casida method, it does not exhibit memory advantages and encounters convergence challenges during actual implementation, even though it is more stable and energy-efficient in theory.

Overall, our optimization schemes offer substantial improvements in computational efficiency and accuracy, paving the way for groundbreaking advancements in the field of LR-TDDFT calculations.

CRediT authorship contribution statement

Qingcai Jiang: Writing – review & editing, Writing – original draft, Software, Resources, Methodology, Formal analysis. **Zhenwei Cao:** Methodology, Investigation, Data curation. **Xinhui Cui:** Formal analysis. **Lingyun Wan:** Software. **Xinming Qin:** Supervision. **Huanqi Cao:** Writing – original draft, Software. **Hong An:** Supervision. **Junshi Chen:** Supervision. **Jie Liu:** Data curation. **Wei Hu:** Writing – original draft, Supervision, Methodology. **Jinlong Yang:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is partly supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (XDB0500102, XDB0450101), the Innovation Program for Quantum Science and Technology (2021ZD 0303306), the National Natural Science Foundation of China (62102 389, 22288201, 22173093, 21688102), the Anhui Provincial Key Research and Development Program (2022a05020052), the National Key Research and Development Program of China (2016YFA0-200604, 2021YFB0300600), and the CAS Project for Young Scientists in Basic Research (YSBR-005). Computing resources of this work are partly financially supported by Laoshan Laboratory (LSKJ202300305).

References

[1] Q. Jiang, J. Li, J. Chen, X. Qin, L. Wan, J. Yang, J. Liu, W. Hu, H. An, Accelerating parallel first-principles excited-state calculation by low-rank approximation with K-means clustering, in: Proceedings of the 51st International Conference on Parallel Processing, 2022, pp. 1–11.
[2] P. Hohenberg, W. Kohn, Inhomogeneous electron gas, Phys. Rev. 136 (3B) (1964) B864.
[3] E. Runge, E.K. Gross, Density-functional theory for time-dependent systems, Phys. Rev. Lett. 52 (12) (1984) 997.
[4] T.L. Beck, Real-space mesh techniques in density-functional theory, Rev. Modern Phys. 72 (4) (2000) 1041.
[5] G. Onida, L. Reining, A. Rubio, Electronic excitations: density-functional versus many-body Green's-function approaches, Rev. Modern Phys. 74 (2) (2002) 601.
[6] J. Sun, C.-W. Lee, A. Kononov, A. Schleife, C.A. Ullrich, Real-time exciton dynamics with time-dependent density-functional theory, Phys. Rev. Lett. 127 (7) (2021) 077401.
[7] L. Jensen, J. Autschbach, G.C. Schatz, Finite lifetime effects on the polarizability within time-dependent density-functional theory, J. Chem. Phys. 122 (22) (2005).
[8] M. Krykunov, M. Seth, T. Ziegler, J. Autschbach, Calculation of the magnetic circular dichroism B term from the imaginary part of the verdet constant using damped time-dependent density functional theory, J. Chem. Phys. 127 (24) (2007).
[9] Z. Hu, J. Autschbach, L. Jensen, Simulating third-order nonlinear optical properties using damped cubic response theory within time-dependent density functional theory, J. Chem. Theory Comput. 12 (3) (2016) 1294–1304.
[10] M.E. Casida, Time-dependent density functional response theory for molecules, in: Recent Advances in Density Functional Methods: (Part I), World Scientific, 1995, pp. 155–192.

[11] J. Strand, S.K. Chulkov, M.B. Watkins, A.L. Shluger, First principles calculations of optical properties for oxygen vacancies in binary metal oxides, J. Chem. Phys. 150 (4) (2019) 044702.
[12] B. Dunlap, J. Connolly, J. Sabin, On first-row diatomic molecules and local density models, J. Chem. Phys. 71 (12) (1979) 4993–4999.
[13] E.R. Davidson, The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices, J. Comput. Phys. 17 (1) (1975) 87–94.
[14] J.A. Duersch, M. Shao, C. Yang, M. Gu, A robust and efficient implementation of LOBPCG, SIAM J. Sci. Comput. 40 (5) (2018) C655–C676.
[15] J. Lu, K. Thicke, Cubic scaling algorithms for RPA correlation using interpolative separable density fitting, J. Comput. Phys. 351 (2017) 187–202.
[16] G. Kresse, J. Furthmüller, Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set, Phys. Rev. B 54 (16) (1996) 11169.
[17] L. Wang, Y. Wu, W. Jia, W. Gao, X. Chi, L.-W. Wang, Large scale plane wave pseudopotential density functional theory calculations on GPU clusters, in: SC'11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2011, pp. 1–10.
[18] T. Zuehlsdorff, P.D. Haynes, F. Hanke, M. Payne, N.D. Hine, Solvent effects on electronic excitations of an organic chromophore, J. Chem. Theory Comput. 12 (4) (2016) 1853–1861.
[19] W. Jia, L.-W. Wang, L. Lin, Parallel transport time-dependent density functional theory calculations with hybrid functional on summit, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2019, pp. 1–23.
[20] M. Del Ben, C. Yang, Z. Li, H. Felipe, S. Louie, J. Deslippe, Accelerating large-scale excited-state GW calculations on leadership HPC systems, in: 2020 SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, SC, IEEE Computer Society, 2020, pp. 36–46.
[21] M. Valiev, E.J. Bylaska, N. Govind, K. Kowalski, T.P. Straatsma, H.J. Van Dam, D. Wang, J. Nieplocha, E. Apra, T.L. Windus, et al., NWChem: A comprehensive and scalable open-source solution for large scale molecular simulations, Comput. Phys. Comm. 181 (9) (2010) 1477–1489.
[22] R. Olivares-Amaya, W. Hu, N. Nakatani, S. Sharma, J. Yang, G.K.-L. Chan, The ab-initio density matrix renormalization group in practice, J. Chem. Phys. 142 (3) (2015) 034102.
[23] E. Apra, E.J. Bylaska, W.A. De Jong, N. Govind, K. Kowalski, T.P. Straatsma, M. Valiev, H.J. van Dam, Y. Alexeev, J. Anchell, et al., NWChem: Past, present, and future, J. Chem. Phys. 152 (18) (2020) 184102.
[24] A.L. Fetter, J.D. Walecka, Quantum theory of many-particle systems, 1971, qtmp.
[25] J. Choi, J.J. Dongarra, R. Pozo, D.W. Walker, ScaLAPACK: A scalable linear algebra library for distributed memory concurrent computers, in: The Fourth Symposium on the Frontiers of Massively Parallel Computation, IEEE Computer Society, 1992, pp. 120–121.
[26] W. Hu, L. Lin, A.S. Banerjee, E. Vecharynski, C. Yang, Adaptively compressed exchange operator for large-scale hybrid density functional calculations with applications to the adsorption of water on silicene, J. Chem. Theory Comput. 13 (3) (2017) 1188–1198.
[27] W. Hu, L. Lin, C. Yang, DGDFT: A massively parallel method for large scale density functional theory calculations, J. Chem. Phys. 143 (12) (2015) 124110.
[28] S. Goedecker, M. Teter, J. Hutter, Separable dual-space Gaussian pseudopotentials, Phys. Rev. B 54 (3) (1996) 1703.
[29] E. Wang, Q. Zhang, B. Shen, G. Zhang, X. Lu, Q. Wu, Y. Wang, Intel math kernel library, in: High-Performance Computing on the Intel® Xeon Phi, Springer, 2014, pp. 167–188.
[30] M. Frigo, S.G. Johnson, FFTW: An adaptive software architecture for the FFT, in: Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181), 3, IEEE, 1998, pp. 1381–1384.
[31] F. Aquilante, T.B. Pedersen, R. Lindh, Low-cost evaluation of the exchange fock matrix from Cholesky and density fitting representations of the electron repulsion integrals, J. Chem. Phys. 126 (19) (2007) 194106.
[32] J. Lu, L. Ying, Compression of the electron repulsion integral tensor in tensor hypercontraction format with cubic scaling cost, J. Comput. Phys. 302 (2015) 329–335.
[33] J.A. Duersch, M. Gu, Randomized QR with column pivoting, SIAM J. Sci. Comput. 39 (4) (2017) C263–C291.
[34] E. Angerson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammarling, J. Demmel, C. Bischof, D. Sorensen, LAPACK: A portable linear algebra library for high-performance computers, in: Supercomputing'90: Proceedings of the 1990 ACM/IEEE Conference on Supercomputing, IEEE, 1990, pp. 2–11.
[35] G.M. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, in: Proceedings of the April 18–20, 1967, Spring Joint Computer Conference, 1967, pp. 483–485.
[36] J.J. Dongarra, J. Du Croz, S. Hammarling, I.S. Duff, A set of level 3 basic linear algebra subprograms, ACM Trans. Math. Softw. 16 (1) (1990) 1–17.
[37] H. Cao, J. Chen, Design and implementation of ShenWei universal C/C++, 2022, arXiv preprint arXiv:2208.00607.
[38] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G.L. Chiarotti, M. Cococcioni, I. Dabo, et al., Quantum Espresso: a modular and open-source software project for quantum simulations of materials, J. Phys.: Condens. Matter. 21 (39) (2009) 395502.
[39] J. Liu, W. Hu, J. Yang, Accelerating linear-response time-dependent hybrid density functional theory with low-rank decomposition techniques in the plane-wave basis, J. Chem. Theory Comput. 18 (11) (2022) 6713–6721.
[40] W. Hu, J. Liu, Y. Li, Z. Ding, C. Yang, J. Yang, Accelerating excitation energy computation in molecules and solids within linear-response time-dependent density functional theory via interpolative separable density fitting decomposition, J. Chem. Theory Comput. 16 (2) (2020) 964–973.
[41] F. Hofmann, I. Schelter, S. Kümmel, Linear response time-dependent density functional theory without unoccupied states: The Kohn-Sham-Sternheimer scheme revisited, J. Chem. Phys. 149 (2) (2018).
[42] H. Hübener, F. Giustino, Time-dependent density functional theory using atomic orbitals and the self-consistent Sternheimer equation, Phys. Rev. B 89 (8) (2014) 085129.
[43] N. Zibouche, M. Schlipf, F. Giustino, GW band structure of monolayer MoS 2 using the SternheimerGW method and effect of dielectric environment, Phys. Rev. B 103 (12) (2021) 125401.