JCST Journal of Computer Science & Technology

Indexed in: SCIE, EI, INSPEC, CBST, DBLP, etc.

Sponsored by: ICT, CAS & CCF

10-Million Atoms Simulation of First-Principle Package LS3DF on Sugon Supercomputer

Yan Yu-Jin, Li Hai-Bo, Zhao Tong, Wang Lin-Wang, Shi Lin, Liu Tao, Tan Guang-Ming, Jia Wei-Le, Sun Ning-Hui

View online: http://doi.org/10.1007/s11390-023-3011-6

Articles you may be interested in

A Large-Scale Study of Failures on Petascale Supercomputers

Rui-Tao Liu, Zuo-Ning Chen

Journal of Computer Science and Technology. 2018, 33(1): 24-41 http://doi.org/10.1007/s11390-018-1806-7

AquaSee: Predict Load and Cooling System Faults of Supercomputers Using Chilled Water Data

Yu-Qi Li, Li-Quan Xiao, Jing-Hua Feng, Bin Xu, Jian Zhang

Journal of Computer Science and Technology. 2020, 35(1): 221-230 http://doi.org/10.1007/s11390-019-1951-7

Ad Hoc File Systems for High-Performance Computing

André Brinkmann, Kathryn Mohror, Weikuan Yu, Philip Carns, Toni Cortes, Scott A. Klasky, Alberto Miranda, Franz-Josef Pfreundt, Robert B. Ross, Marc-André Vef

Journal of Computer Science and Technology. 2020, 35(1): 4-26 http://doi.org/10.1007/s11390-020-9801-1

Mochi: Composing Data Services for High-Performance Computing Environments

Robert B. Ross, George Amvrosiadis, Philip Carns, Charles D. Cranor, Matthieu Dorier, Kevin Harms, Greg Ganger, Garth Gibson, Samuel K. Gutierrez, Robert Latham, Bob Robey, Dana Robinson, Bradley Settlemyer, Galen Shipman, Shane Snyder, Jerome Soumagne, Qing Zheng

Journal of Computer Science and Technology. 2020, 35(1): 121-144 http://doi.org/10.1007/s11390-020-9802-0

Distinguishing Computer-Generated Images from Natural Images Using Channel and Pixel Correlation

Rui-Song Zhang, Wei-Ze Quan, Lu-Bin Fan, Li-Ming Hu, Dong-Ming Yan

Journal of Computer Science and Technology. 2020, 35(3): 592-602 http://doi.org/10.1007/s11390-020-0216-9

Usage Scenarios for Byte-Addressable Persistent Memory in High-Performance and Data Intensive Computing

Michèle Weiland, Bernhard Homlle

Journal of Computer Science and Technology. 2021, 36(1): 110-122 http://doi.org/10.1007/s11390-020-0776-8

JCST Homepage: https://jcst.ict.ac.cn
SPRINGER Homepage: https://www.springer.com/journal/11390
E-mail: jcst@ict.ac.cn
Online Submission: https://mc03.manuscriptcentral.com/jcst

JCST Official
WeChat Account

JCST WeChat
Service Account

Twitter: JCST_Journal
LinkedIn: Journal of Computer Science and Technology

Yan YJ, Li HB, Zhao T *et al.* 10-million atoms simulation of first-principle package LS3DF on Sugon supercomputer. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 39(1): 45-62 Jan. 2024. DOI: 10.1007/s11390-023-3011-6

10-Million Atoms Simulation of First-Principle Package LS3DF on Sugon Supercomputer

Yu-Jin Yan[1, 2] (严昱瑾), *Student Member, CCF*, Hai-Bo Li[1, 3, *] (李海波), Tong Zhao[2] (赵　曈), *Member, CCF*
Lin-Wang Wang[4] (汪林望), Lin Shi[5] (石　林), Tao Liu[1] (刘　涛), *Member, CCF*
Guang-Ming Tan[1] (谭光明), *Member, CCF, ACM, IEEE*, Wei-Le Jia[1, *] (贾伟乐), *Member, CCF, ACM*
and Ning-Hui Sun[1, 2, *] (孙凝晖), *Fellow, CCF, Member, ACM, IEEE*

[1] *State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190 China*

[2] *University of Chinese Academy of Sciences, Beijing 101408, China*

[3] *Computing System Optimization Laboratory, Huawei Technologies, Beijing 100094, China*

[4] *Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China*

[5] *School of Materials Science and Engineering, Yancheng Institute of Technology, Yancheng 224051, China*

E-mail: yanyujin17z@ict.ac.cn; lihaibo39@huawei.com; zhaotong@ict.ac.cn; lwwang@semi.ac.cn; linshi@ycit.edu.cn
liutao17@ict.ac.cn; tgm@ict.ac.cn; jiaweile@ict.ac.cn; snh@ict.ac.cn

Received February 21, 2023; accepted April 25, 2023.

Abstract The growing demand for semiconductor devices simulation poses a big challenge for large-scale electronic structure calculations. Among various methods, the linearly scaling three-dimensional fragment (LS3DF) method exhibits excellent scalability in large-scale simulations. Based on algorithmic and system-level optimizations, we propose a highly scalable and highly efficient implementation of LS3DF on the Sugon supercomputer, a domestic supercomputer equipped with deep computing units. In terms of algorithmic optimizations, the original all-band conjugate gradient algorithm is refined to achieve faster convergence, and mixed precision computing is adopted to increase overall efficiency. In terms of system-level optimizations, the original two-layer parallel structure is replaced by a coarse-grained parallel method. Optimization strategies such as multi-stream, kernel fusion, and redundant computation removal are proposed to increase further utilization of the computational power provided by the heterogeneous machines. As a result, our optimized LS3DF can scale to a 10-million silicon atoms system, attaining a peak performance of 34.8 PFLOPS (21.2% of the peak). All the improvements can be adapted to the next-generation supercomputers for larger simulations.

Keywords deep computing unit, electronic structure, high-performance computing, linearly scaling three-dimensional fragment (LS3DF), Sugon supercomputer

1 Introduction

The exponential increase of the computing power described by Moore's law has been steadily driven by fundamental advances in material sciences. To date, semiconductor devices such as field-effect transistors (FETs) are the cornerstone of integrated circuits (IC) and the entire information industry. As the size of FETs shrinks to less than 10 nm[1], quantum mechanics phenomena (electronic structure, band gap open-

Regular Paper

This work was supported by the National Key Research and Development Program of China under Grant No. 2021YFB0300600, the National Natural Science Foundation of China under Grant Nos. 92270206, T2125013, 62032023, 61972377, T2293702, and 12274360, the Chinese Academy of Sciences Project for Young Scientists in Basic Research under Grant No. YSBR-005, the Network Information Project of Chinese Academy of Sciences under Grant No. CASWX2021SF-0103, and the Key Research Program of Chinese Academy of Sciences under Grant No. ZDBSSSW-WHC002.

*Corresponding Author (Hai-Bo Li is responsible for algorithm design and participated in paper writing; Wei-Le Jia is responsible for the overall design and guidance of the paper work, and algorithmic optimization; Ning-Hui Sun is the chief instructor of the work and responsible for system optimization.)

ing, band alignment, and charge transfer[2]) are playing a more essential role in modeling next-generation semiconductor devices. To model the quantum effects, the governing equation of quantum mechanics such as Kohn-Sham Density Functional Theory (KS-DFT)[3] is evaluated. One fundamental challenge is the computational and memory requirement of large-scale problems on high-performance supercomputers. The computational cost of DFT scales cubically with respect to the system size, which leads to that most practical DFT softwares such as the plane wave methods[4, 5] and finite elements methods[6–9] can rarely reach a few thousands of atoms on current supercomputers[10]. Only with the development of low complexity and scalable new algorithms in the past decade, it is now possible to carry out DFT calculation for systems with tens of thousands of atoms. However, for a silicon FET of 100 nm×40 nm×7 nm, the system size can easily reach 1 million atoms.

Many efforts are endeavored to model large-scale materials from first-principles calculations, as listed in Table 1. One notable point is the development of the conventional cubic scaling methods. For example, Gygi *et al.*[11] calculated a Mo system of 1k atoms in 2006 with QBox, and DFT-FE[10] reaches up to 11k atoms on Summit in 2019, attaining a peak performance of 46 PFLOPS. Note that the system size increases by a factor of 11 from 2006 to 2019, while the theoretical peak of the top supercomputers increased by a factor of 550. This nearly aligns with the cubic scaling curve of the conventional methods. However, to reach the size of a typical semiconductor device of 1 million atoms, the corresponding floating point operations of individual self-consistent filed calculation can reach the order of EFLOPS, which is already beyond the scope of the top supercomputers available.

Besides the conventional cubic scaling methods, low-scaling methods such as the linearly scaling and Fermi operator expansion (FOE) methods[20–22] are more favorable in effectively calculating large-scale calculations with *ab initio* accuracy. The RSDFT code[13] uses the real-space finite-difference method where the matrix of real-space formulation is sparse and fast Fourier transformation (FFT) is unnecessary for Hamiltonian matrix operations, which provides a great advantage by easing the communication burden in parallel computations; it can calculate electronic states in a vast range of physical systems including crystals, interfaces, molecules, etc. The CP2K code[15] uses a plane wave auxiliary basis set within a Gaussian orbital scheme that sets it apart from most other DFT softwares, and the linearly scaling computational complexity benefits from the use of Spatially localized molecular orbitals; it can perform atomistic simulations of solid state, liquid, molecular, periodic, material, crystal, and biological systems. CON-

Table 1. Performance Comparison of Massively Parallel DFT Software Packages on Modern Heterogeneous Supercomputers

Code	Year	Basis	Eigensolver	System	Number of Atoms	Machine	Architecture	Scale	Peak (FLOPS)
Qbox[11]	2006	PW	Davidson	Mo	1k	BlueGene/L	PowerPC	128k cores	207T
LS3DF[12]	2008	PW	LS	ZnTeO	16k	BlueGene/R	PowerPC	131k cores	108T
RSDFT[13]	2011	RS	LS	Si	107k	K computer	SPARC64	442k cores	3.1P
DFT-FE[10]	2019	RS-PW	CheFSI	Mg	11k	Summit	V100s	23k GPUs	46P
CONQUEST[14]	2020	NAOs	LS	Si	1M	K computer	SPARC64	200k cores	\
CP2K[15]	2020	GAPW	LS	H_2O	1k	PC^2	Noctua	10k cores	\
FHI-aims[16]	2021	NAOs	LS	Polyethylene ($H[C_2H_4]_nH$)	500k	New Sunway	sw26010 pro	40M cores	468.5P
DGDFT[17]	2021	DG-PW	CheFSI	Graphene	13k	Sunway TaihuLight	sw26010	8.5M cores	\
NOLSM[18]	2022	GTO	LS	Bulk water	102M	JUWELS Booster	A100s	1.5k GPUs	106P
DGDFT[19]	2022	DG-PW	CheFSI	MATBG, Li/Na, Cu/G/Cu, LAO/STO	2.5M	Sunway	sw26010 pro	40M cores	64P
Optimized LS3DF (this work)	2022	PW	LS	Si	10M	Sugon	Z100SM	14.4k DCUs	34.8P

Note: The DFT methods include cubic-scaling plane-wave (PW) and localized real-space (RS) basis sets, numerical atomic orbitals (NAOs), linear-scaling (LS) solvers, Gaussian augmented plane wave orbitals (GAPW) and Gaussian-type orbitals (GTO). Qbox adopts cubic-scaling conjugate gradient eigensolvers (Davidson). DGDFT and DFT-FE can use CheFSI eigensolver. LS3DF, RSDFT, CONQUEST, FHI-aims, and NOLSM exploit LS eigensolvers.

QUEST[14] uses the NAOs basis and it is a linear-scaling DFT code based on the density matrix minimization method that can employ structure optimization or molecular dynamics on very large-scale systems including more than millions of atoms. The FHI-aims software[16] is an all-electron, full-potential electronic structure code utilizing NAOs, and it can achieve all-electron accuracy at a computational cost comparable to plane-wave/pseudopotential implementations on large-scale system of hundreds of thousands atoms thanks to its linearly scaling computation cost. Recently, the discontinuous Galerkin density functional theory (DGDFT) method implements a highly efficient pole expansion and selected inversion (PEXSI) sparse direct solver to achieve a 2.5 million atoms metallic heterostructure simulation on the new Sunway supercomputer[19]. Among these various linear scaling methods, the linearly scaling three-dimensional fragment (LS3DF) method proposed by Wang *et al.*[12, 23] exploits a smart divide-and-conquer approach for large-scale systems, and is particularly applicable for large-scale simulations of insulator and semiconductor systems that exhibit excellent scalability, i.e., 16k-atom ZnTeO system is calculated on BlueGene/R (Table 1). Furthermore, the recent development of the LS3DF method makes it possible to model copper devices with 5 000 atoms[24].

The recent development of domestic supercomputers built from in-house hardware such as Sugon, Sunway, and Tianhe supercomputers lays a solid foundation for modeling nanometer devices, e.g., calculating physical systems beyond million atoms. One crucial issue is the increasing requirements for better parallel performance and scalability of the implementation in order to achieve large-scale parallel DFT calculations. Previous attempts to push the boundaries of DFT calculations are summarized in Table 1. For example, the massively parallel implemented DGDFT method on the Sunway TaihuLight supercomputer adopts a two-level parallelization strategy that makes use of different types of data distribution, task scheduling, and data communication schemes, which finally achieves the DFT calculation for tens of thousands of atoms[17]. To achieve linear scaling and simulate large-scale systems, LS3DF uses a divide-and-conquer approach where the system is spatially divided into small pieces and each piece can be solved independently by a small group of processors. The crux of this algorithm is a novel patching scheme that cancels out the artificial boundary effects caused by the division of the system into smaller fragments. Up to now, however, there is still a gap between the current implementation of LS3DF and the state-of-the-art DFT calculation software. The previous GPU-accelerated LS3DF is performed on heterogeneous machines[25], where the data communication task takes a large amount of time, leading to relatively low efficiency of the implementation.

In order to achieve much better implementation of LS3DF, there are three main challenges. The first is the need for a faster process for the eigenvalue decomposition of Hamiltonian matrices, which is the most time-consuming part of the whole calculation. The all-band conjugate gradient (AB_CG) algorithm[26], which is proposed for LS3DF to approximate all desired occupied orbitals simultaneously, still has room to refine to obtain a faster convergence. Meanwhile, the availability of low-precision floating point formats on Sugon's deep computing units (DCUs) should also be considered to save the cost of communications and computations. The second is the two-layer parallel structure of LS3DF, designed for the small memory of previous GPUs but resulting in a substantial amount of data movement overhead. However, considering the memory advantage of high-paralleled DCUs over those GPUs, this structure is not suitable for our heterogeneous supercomputer. Finally, there are some other computational tasks with lower computation efficiency in DCUs, such as repeated use of Fourier transform and inverse Fourier transform, multiplication, addition, and decomposition of matrices. Thus some further strategies are needed to enhance computation efficiency and increase DCUs utilization.

In this paper, we focus on the implementation of a high-performance and highly scalable LS3DF code on the heterogeneous Sugon platform to overcome the above challenges. To this end, both algorithmic and system-level optimization have to be employed. For algorithmic optimization, we modify the original AB_CG algorithm to get a faster convergence of approximate eigenvalues and eigenvectors, which can lead to a faster convergence of the self-consistent field (SCF) iteration. We also use mixed-precision computing strategies in the AB_CG to make full use of DCUs' support for lower-precision computing. It can be seen from both theoretical analysis and numerical results that these algorithmic optimizations do increase computational efficiency while maintaining the same computational correctness as the original implementation. For system-level optimization, the origi-

nal two-layer parallel architecture, which is composed of task parallelism in the first layer and data parallelism in the second layer, is optimized using course-grained parallelism to decrease data transfer and speed up computations. Moreover, we propose a multi-stream 3D FFT approach, while kernel fusion and redundant computation removal are also exploited to increase further DCUs' efficiency.

Numerical results on an 8 000 silicon atoms system show that the computation time of our optimized code can be three times faster compared with the previous implementation. Scalability tests show that our algorithm can maintain 81% efficiency in strong scalability from 375 nodes to 3 000 nodes which can scale to a 10 million silicon system, attaining a nearly perfect scaling and 21.2% of peak performance of 34.8 PFLOPS on a domestic Sugon supercomputer with 3 800 nodes. All these improvements can be transformed into next-generation supercomputers for larger system simulations, which will play an important role in modeling next-generation semiconductor devices.

2 LS3DF Method

The LS3DF method is used in large-scale *ab initio* molecular dynamics (AIMD) for calculating the ground state total energy of a material system. It adopts a divide-and-conquer strategy to break a large system into several fragments and get the total energy of the whole system by solving each fragment. These two parts are discussed in more detail below.

2.1 System Division

By the Kohn-Sham density functional theory, the total energy can be divided into quantum mechanical and classical electrostatic components[27]. The electrostatic component (Coulomb energy) for large systems can be easily solved by using Poisson solvers due to its long-range interaction nature, while the quantum mechanical part (the kinetic energy and exchange-correlation energy) is computationally complex and very time-consuming. Fortunately, this problem can be handled by calculating energy locally due to the short-range interaction property. LS3DF uses a smart strategy to divide the system into several fragments to calculate their local quantum mechanical energies independently and then sum them together. As a result, the total quantum mechanical energy can be determined more quickly with $O(N)$ computational scaling while obtaining the same original full-system DFT computed results.

The critical issue in this method is how to divide the whole system into fragments and put the fragments together without introducing artificial boundary effects, which can be achieved by using the following special division and patching scheme method illustrated in Fig.1. For ease of comprehension, we use a two-dimensional system as an illustration here. The case for three-dimensional systems is similar. First, the system is divided into $m_1 \times m_2$ small pieces (fragments). For all fragments (i, j), we calculate the quantum energy and charge density of them, which are the orange (size 1×1), the blue (size 1×2), the green (size 2×1) and the yellow (size 2×2) squares, respectively. Quantum energy and charge density are calculated as $E_{i,j,S}$ and $\rho_{i,j,S}$, respectively, where S represents different fragment sizes. Then the total quantum energy of the system can be calculated as $E = \Sigma_{i,j,S}\alpha_S E_{i,j,S}$, and the total charge density as $\rho_{\text{tot}}(\boldsymbol{r}) = \Sigma_{i,j,S}\alpha_S \rho_{i,j,S}(\boldsymbol{r})$, where α_S is the sign of different fragments with $\alpha_S = 1$ if the fragment size $S = 1 \times 1$ or 2×2 and $\alpha_S = -1$ if the fragment size $S = 2 \times 1$ or 1×2. In this summation process, the quantum energy and total charge density cover precisely the entire area of the whole system (both on the inside and at the edges), and the boundary effects and corner effects can be canceled out. A more detailed explanation can be found in [12]. Next, the electron-electron Coulomb energy is calculated by using the total charge density $\rho_{\text{tot}}(\boldsymbol{r})$. Finally, the quan-

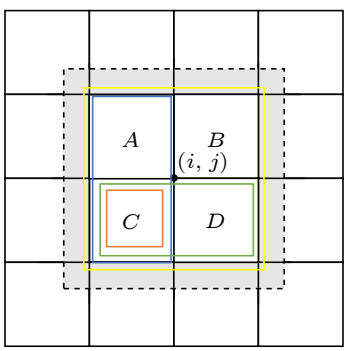


Fig.1. Division of space and fragment pieces. Fragment (i, j) corresponds to the result of the calculation of square B. The figure shows the shapes and sizes of the regions involved in the calculation of fragment (i, j), where the regions of squares A, B, C and D are needed to be involved in the calculation. The orange squares represent the region of square C, the blue squares represent the regions of squares A and C, the green squares represent the regions of squares C and D, and the yellow squares represent the regions of squares A, B, C and D.

tum energy and Coulomb energy are summed together to get the total energy.

2.2 Fragment Conquest

By using the above division method, the quantum energy of each fragment can be calculated independently. The quantum energy can be determined by the self-consistent field (SCF) iteration, which involves five main steps. Subsection 2.2.1 will describe each step in more detail. One of the most time-consuming steps is solving the eigenvalue problem in the Kohn-Sham equation, and we describe how to solve it using the all-band conjugate gradient (AB_CG) algorithm in Subsection 2.2.2.

2.2.1 *Self-Consistent Field Iteration (SCF)*

The SCF iteration is illustrated in Fig.2. It is used to solve the global Kohn-Sham equation:

$$\left(-\frac{1}{2}\nabla^2 + V_{\text{ef}}(\rho_{\text{ef}}(\boldsymbol{r}))\right)\boldsymbol{\Psi}_i(\boldsymbol{r}) = E_i\boldsymbol{\Psi}_i(\boldsymbol{r}),$$

with charge density $\rho_{\text{ef}}(\boldsymbol{r}) = \sum_{i=1}^{N}|\boldsymbol{\Psi}_i(\boldsymbol{r})|^2$ by a divide-and-conquer strategy. There are five steps in each iteration. At each SCF iteration, starting from an initial total potential $V_{\text{in}}^{\text{tot}}(\boldsymbol{r})$, the GEN_VF part generates the potentials of each fragment $V_{\text{F}}(\boldsymbol{r}) = V_{\text{in}}^{\text{tot}}(\boldsymbol{r}) + \Delta V_{\text{F}}(\boldsymbol{r})$ based on the above division method. Next, PEtot_F solves the eigenvalue problem $((-\nabla^2/2) + V_{\text{F}}(\boldsymbol{r}))\Psi_i^{\text{F}}(\boldsymbol{r}) = E_i^{\text{F}}\Psi_i^{\text{F}}(\boldsymbol{r})$ on each fragment

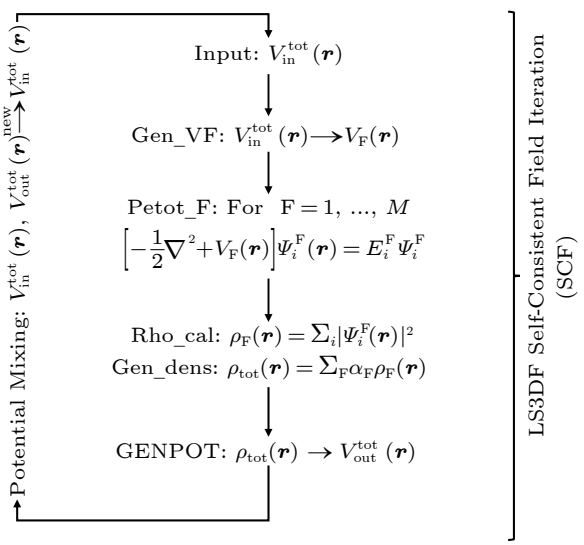


Fig.2. LS3DF self-consistent field flow chart.

to obtain approximate wave functions $\Psi_i^{\text{F}}(\boldsymbol{r})$. The AB_CG algorithm is used to accomplish this, which will be described in Subsection 2.2.2. After obtaining the approximate wave functions $\Psi_i^{\text{F}}(\boldsymbol{r})$ on each fragment, the fragment charge density $\rho_{\text{F}}(\boldsymbol{r}) = \Sigma_i|\Psi_i^{\text{F}}(\boldsymbol{r})|^2$ is calculated. According to the subroutine Gen_dens, the approximate charge density of the total system is $\rho_{\text{tot}}(\boldsymbol{r}) = \Sigma_{i,j,S}\alpha_S\rho_{i,j,S}(\boldsymbol{r})$. In the fourth step, we use $\rho_{\text{tot}}(\boldsymbol{r})$ to determine the global potential $V_{\text{out}}^{\text{tot}}(\boldsymbol{r})$ through the global Poisson solver GENPOT. In the final step, the potential of the current step is mixed with the potential from the previous step, and the updated potential will be used as the input potential for the next iteration. As the iterations progress, the SCF solutions will converge to the true charge density $\rho_{\text{ef}}(\boldsymbol{r})$ and Kohn-Sham potential $V_{\text{ef}}(\rho_{\text{ef}}(\boldsymbol{r}))$, and then the ground state energy can be obtained.

2.2.2 *All-Band Conjugate Gradient Algorithm*

Solving the eigenvalue problem $\boldsymbol{H}\boldsymbol{\Psi}_i = \varepsilon_i\boldsymbol{\Psi}_i$ is the most computationally intensive part, accounting for up to 90% of the total execution time. Many methods, such as direct solver($O(N^3)$), selected inversion($O(N^{1-2})$), and iterative solvers($O(N^{2-3})$[4, 21, 28], are introduced to effectively solve this problem on high-performance platforms. Among them, iterative methods based on conjugate gradient method are known to be both stable and effective, and especially, high-performance libraries such as GEMM and FFT can be easily adopted when plane wave discretization is used in software packages such as LS3DF. The (single-band) conjugate gradient method[4] is by Payne *et al.* Then several CG variants, for example, blocked CG[29], projected preconditioned CG[30], and locally optimal block preconditioned CG[31], are adopted in different software. Note that these CG methods differ from each other in their stability, convergence, and scalability. In this paper, we use an AB_CG method[32][①], which differs from all the above CG methods by simultaneously updating all wavefunctions using GEMM operations. Therefore, AB_CG can easily utilize the enormous computing power provided by the many-core architecture, such as DCUs used in this paper. AB_CG also has a relatively small subspace $(N_e \times N_e)$ instead of a big subspace like LOBPCG $(3N_e \times 3N_e)$ to reduce the MPI_Allreduce communication across all MPI ranks. It is stable and

[①]https://github.com/qsnake/petot, July 2023.

converges fast for many physical problems. Overall it is a good trade-off among computation, data movement, stability, and convergency. Therefore, it has been implemented in several software packages, such as PWmat, PEtot, and LS3DF[12, 25, 32].

3 Innovation Implementation

3.1 Summary of Contributions

The most important contribution of this paper is the implementation of a high-performance and highly scalable LS3DF code on a heterogeneous HPC platform. To this end, both algorithmic and system-level optimization have been employed. Our optimized version of LS3DF can scale perfectly up to 3 800 computing nodes, extending 10 million atoms, and reaching a peak performance of 34.8 PFLOPS.

3.2 Algorithmic Innovations

For algorithmic optimization, we propose two approaches, including improving the convergence of AB_CG by choosing the optimal angle $\theta_i^{(k)}$ at each iteration k, and using mixed precision for computations. In light of the following analysis, it can be seen that our two optimizations can maintain the same computational correctness as the original implementation. This is also confirmed by the results of numerical experiments.

3.2.1 *Optimal Angle in the AB_CG Algorithm*

The basic intuition of the AB_CG algorithm is to minimize the objective functional

$$E[\boldsymbol{\Psi}] = \mathrm{Trace}\langle \boldsymbol{\Psi} \mid \boldsymbol{H} \mid \boldsymbol{\Psi}\rangle = \sum_{i=1}^{N} \langle \boldsymbol{\Psi}_i \mid \boldsymbol{H} \mid \boldsymbol{\Psi}_i\rangle,$$

with orthonormal wave functions $\boldsymbol{\Psi} = (\boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_N)$. For simplicity of notations, in the following discussion we omit the superscript "(k)" for every computed quantity. The all-band method uses the conjugate gradient algorithm on each $\langle \boldsymbol{\Psi}_i \mid \boldsymbol{H} \mid \boldsymbol{\Psi}_i\rangle$ to get all the minimizers simultaneously, where at each step the method updates $\boldsymbol{\Psi}_i$ as line 15 uses a carefully chosen angle θ_i, which should be the minimizer of

$$\begin{aligned}\varphi(\theta) &= \langle \cos\theta\boldsymbol{\Psi}_i + \sin\theta\boldsymbol{P}_i \mid \boldsymbol{H}(\cos\theta\boldsymbol{\Psi}_i + \sin\theta\boldsymbol{P}_i)\rangle \\ &= \lambda_i \cos^2\theta + \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle \sin^2\theta + \langle \boldsymbol{P}_i \mid \boldsymbol{\Phi}_i\rangle \sin(2\theta),\end{aligned}$$

where $\boldsymbol{\Phi}_i = \boldsymbol{H}\boldsymbol{\Psi}_i$.

We now show how to compute the optimal angle θ_i. By line 11, P_i is orthogonal to Ψ_i, and thus the updated Ψ_i in line 11 is of unit length. The minimizer of $\varphi(\theta)$ satisfies $\varphi'(\theta) = 0$. Note that

$$\varphi'(\theta) = -(\lambda_i - \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle)\sin(2\theta) + 2\langle \boldsymbol{P}_i \mid \boldsymbol{\Phi}_i\rangle \cos(2\theta),$$

which leads to

$$\tan(2\theta) = \frac{2\langle \boldsymbol{P}_i \mid \boldsymbol{\Phi}_i\rangle}{\lambda_i - \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle} = \frac{2\langle \boldsymbol{\Phi}_i \mid \boldsymbol{\Theta}_i\rangle}{\lambda_i - \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle}.$$

By solving the above equation we get two $\theta \in [0, \pi]$, one minimizes $\varphi(\theta)$ and the other maximizes $\varphi(\theta)$, and the two values of θ differ by $\pi/2$. In order to obtain the minimizer, we compute

$$\begin{aligned}\varphi''(\theta) &= -2[\lambda_i - \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle]\cos(2\theta) - 4\langle \boldsymbol{\Phi}_i \mid \boldsymbol{\Theta}_i\rangle \sin(2\theta) \\ &= -2[\lambda_i - \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle]\cos(2\theta)[1 + \tan^2(2\theta)].\end{aligned}$$

Therefore, the minimizer satisfies $[\lambda_i - \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle]\cos(2\theta) < 0$. By analyzing the sign of $\lambda_i - \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle$ and $\langle \boldsymbol{\Phi}_i \mid \boldsymbol{\Theta}_i\rangle$ we get the minimizer (may be differ by π)

$$\theta_i = \begin{cases} \dfrac{1}{2}\arctan\dfrac{2\langle \boldsymbol{\Phi}_i \mid \boldsymbol{\Theta}_i\rangle}{\lambda_i - \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle} + \dfrac{\pi}{2}, \\ \quad \text{if } \lambda_i - \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle > 0, \\ \dfrac{1}{2}\arctan\dfrac{2\langle \boldsymbol{\Phi}_i \mid \boldsymbol{\Theta}_i\rangle}{\lambda_i - \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle}, \\ \quad \text{if } \lambda_i - \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle < 0. \end{cases} \tag{1}$$

In the original implementation of AB_CG, θ_i is obtained by simply choosing

$$\theta_i = \frac{1}{2}\left|\arctan\frac{2\langle \boldsymbol{\Psi}_i \mid \boldsymbol{\Theta}_i\rangle}{\langle \boldsymbol{\Psi}_i \mid \boldsymbol{\Phi}_i\rangle - \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle}\right|.$$

Although it can save a few computations by neglecting judging the sign of $\lambda_i - \langle \boldsymbol{P}_i \mid \boldsymbol{\Theta}_i\rangle$, this simply chosen angle may deviate from the optimal angle with the value at most $\pi/2$, which results in a lower convergence rate of each minimizer $\boldsymbol{\Psi}_i$. On the contrary, using the optimal angle instead of the original crudely chosen one can really accelerate the convergence of the SCF iteration, as to be shown in the numerical results section (Section 6).

3.2.2 *Mixed Precision Computations*

In order to further optimize the performance of the AB_CG algorithm, we use the mixed precision strategy for computations. Over the past few years, numerous studies have explored the application of mixed precision within this field. For instance, Das *et al.*[10] employed single precision to diminish communi-

cation, while Fattebert *et al.*[33] utilized single precision directly for wave function representation. In Algorithm 1, we eradicate all communication by implementing coarse-grained parallelism in the latter part. However, when using single-precision representation for the wave function, our algorithm is unable to preserve the accuracy of the computational outcomes. Consequently, we have endeavored to develop a novel mixed-precision approach to address this limitation. Notice that the most time-consuming step in Algorithm 1 is the calculation of $\boldsymbol{\Theta}_i = \boldsymbol{H}\boldsymbol{P}_i$ in line 9. Thus, we perform the matrix-vector multiplication of $\boldsymbol{H}\boldsymbol{P}_i$ with single precision, while all of the other computations are carried out with double precision, that is, the storage formats of $\boldsymbol{H}$ and $\boldsymbol{P}_i$ are first converted into single precision, then $\boldsymbol{\Theta}_i = \boldsymbol{H}\boldsymbol{P}_i$ is computed with single precision and finally $\boldsymbol{\Theta}_i$ is stored with double precision.

Algorithm 1. AB_CG Algorithm

Input: $\boldsymbol{H}$, initial vectors $\boldsymbol{\Psi}^{(0)} = (\boldsymbol{\Psi}_1^{(0)} \dots, \boldsymbol{\Psi}_n^{(0)})$, Teter preconditioner $\boldsymbol{M}$

Output: approximate eigenvalues ε_i and corresponding eigenvectors $\boldsymbol{\Psi}_i$ for $i = 1, \dots, n$

1: $[\varepsilon_i^{(0)}, \boldsymbol{\Psi}_i^{(0)}] = \text{diagonalize}(\langle \boldsymbol{\Psi}^{(0)} \mid \boldsymbol{H} \mid \boldsymbol{\Psi}^{(0)} \rangle)$ ▷Sub_diag

2: $\boldsymbol{\Phi}^{(0)} = \boldsymbol{H}\boldsymbol{\Psi}^{(0)}$ ▷ *Hpsi*

3: **for** $k = 0, \dots, N-1$ **do**

4: $\lambda_i^{(k)} = \langle \Psi_i^{(k)} \mid \Phi_i^{(k)} \rangle$, $i = 1, \dots, n$ ▷ $\boldsymbol{\Psi}_i^{(k)}$ is the i-th approximate wave function at the k-th iteration

5: $\boldsymbol{R}^{(k)} = \Phi^{(k)} - \boldsymbol{\Psi}^{(k)} \text{diag}(\lambda_1^{(k)}, \dots, \lambda_n^{(k)})$

6: **if** $k = 0$ **then**

7: $\boldsymbol{P}_i^{(k)} = -\boldsymbol{R}_i^{(k)}$, $i = 1, \dots, n$

8: **else**

9: $\boldsymbol{P}_i^{(k)} = -\boldsymbol{M}\left(\boldsymbol{R}_i^{(k)} - \dfrac{\beta_1}{\beta_0}\boldsymbol{P}_i^{(k-1)}\right)$, $\beta_0 = \langle \boldsymbol{P}_i^{(k-1)} \mid \boldsymbol{P}_i^{(k-1)} \rangle$, $\beta_1 = \langle \boldsymbol{R}_i^{(k)} \mid \boldsymbol{R}_i^{(k)} \rangle$, $i = 1, \dots, n$

10: **end if**

11: $\boldsymbol{P}^{(k)} = \boldsymbol{P}^{(k)} - \boldsymbol{\Psi}^{(k)} \langle \boldsymbol{\Psi}^{(k)} \mid \boldsymbol{P}^{(k)} \rangle$ ▷ Projection

12: $\boldsymbol{\Theta}^{(k)} = \boldsymbol{H}\boldsymbol{P}^{(k)}$ ▷ *Hpsi*

13: **for** $i = 1, \dots, n$ **do** ▷ Can be computed simultaneously

14: Compute the optimal $\theta_i^{(k)}$ by (1)

15: $\boldsymbol{\Psi}_i^{(k+1)} = \boldsymbol{\Psi}_i^{(k)} \cos\theta_i^{(k)} + \boldsymbol{P}_i^{(k)} \sin\theta_i^{(k)}$

16: $\boldsymbol{\Phi}_i^{(k+1)} = \boldsymbol{\Phi}_i^{(k)} \cos\theta_i^{(k)} + \boldsymbol{\Theta}_i^{(k)} \sin\theta_i^{(k)}$

17: **end for**

18: $[\boldsymbol{R}] = \text{Cholesky}(\langle \boldsymbol{\Psi}^{(k+1)} \mid \boldsymbol{\Psi}^{(k+1)} \rangle)$

19: $\boldsymbol{\Psi}^{(k+1)} = \boldsymbol{\Psi}^{(k+1)} \boldsymbol{R}^{-1}$ ▷ Orthogonalization

20: $\boldsymbol{\Phi}^{(k+1)} = \boldsymbol{\Phi}^{(k+1)} \boldsymbol{R}^{-1}$

21: **end for**

22: $[\varepsilon_i, \boldsymbol{\Psi}_i] = \text{diagonalize} \langle \boldsymbol{\Phi}^{(N)} \mid \boldsymbol{\Phi}^{(N)} \rangle$ ▷ Sub_diag

In order to give a theoretical analysis of the mixed precision AB_CG algorithm with the end to show that the single precision computing of $\boldsymbol{H}\boldsymbol{P}_i$ will not sacrifice the accuracy of the final result, we need to assume that all computations are performed in exact arithmetic except for $\boldsymbol{H}\boldsymbol{P}_i$. Using plane wave basis, with an appropriate discretization scheme, a single electron wave function can be represented by a vector $\boldsymbol{x}_i \in \mathbb{C}^n$, where n is the spatial degrees of freedom, i.e., the number of basis functions on a fragment, and the Hamiltonian is represented by an Hermitian matrix $\boldsymbol{A} \in \mathbb{C}^{n \times n}$. In order to simplify the proof, we focus on the case that $\boldsymbol{A}$ is real symmetric and $\boldsymbol{x}_i \in \mathbb{R}^n$. The following proof can be applied to the complex number case with little adjustment since an Hermitian matrix shares almost the same properties as a real symmetric one. We use $\|\cdot\|_2$ and $\|\cdot\|_{\text{F}}$ to denote the 2-norm and Frobenius norm of a matrix or vector, respectively.

Theorem 1. *Suppose the practically needed charge density $\hat{\rho}$ to form the Hamiltonian has accuracy ε with respect to the exact ρ, i.e.,*

$$\|\hat{\rho}-\rho\|_2 \leqslant \varepsilon.$$

The roundoff unit in single precision is denoted by u. If

$$u \ll \frac{\lambda_{N+1}-\lambda_N}{\|\boldsymbol{A}\|_F}\varepsilon, \tag{2}$$

where λ_i is the i-th smallest eigenvalue of $\boldsymbol{A}$, then the single precision computation of $\boldsymbol{A}\boldsymbol{x}_i$ will not influence the solution accuracy of the SCF iteration.

Proof. By the Kohn-Sham density functional theory, H is a functional of charge density ρ, which can be represented by

$$\rho(\boldsymbol{X}) = \mathrm{diag}(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}), \tag{3}$$

where $\boldsymbol{X} = (x_1, \ldots, x_N) \in \mathbb{R}^{n\times N}$ satisfying $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} = \boldsymbol{I}_N$ is the approximation of N occupied electron wave functions after the discretization, and $\boldsymbol{P}(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$ is the density matrix[29]. At each SCF iteration, the matrix $\boldsymbol{A}$ is dependent on $\rho(\boldsymbol{X})$ and thus $\boldsymbol{X}$, which is composed of the orthonormal basis of the eigenspace corresponding to the N smallest eigenvalues of the discretized Hamiltonian in the previous iteration. Therefore, we only need to prove that the computed version of $\rho(\boldsymbol{X})$ has the same accuracy as $\hat{\rho}$.

Let $\hat{y} = \mathrm{fl}(\boldsymbol{A}x)$, where $\mathrm{fl}(\cdot)$ denotes the computed quantity in finite precision arithmetic. Then we have the backward error result:

$$\hat{y} = (\boldsymbol{A}+\boldsymbol{E})x, \quad \|\boldsymbol{E}\|_{\mathrm{F}} = O(\|\boldsymbol{A}\|_{\mathrm{F}}u).$$

$\boldsymbol{E}$ is an error matrix (referring to Subsection 3.5 of [34]) that can be treated as a symmetric one. Since the AB_CG is carried out in exact arithmetic except for the computation of $\boldsymbol{A}x$ at each iteration, the algorithm actually approximates eigenvalues and eigenvectors of the symmetric $\boldsymbol{A}+\boldsymbol{E}$. We denote by $\hat{\boldsymbol{X}}$ the matrix composed by the orthonormal basis of the eigenspace corresponding to the N smallest eigenvalues of $\boldsymbol{A}+\boldsymbol{E}$. Let $\boldsymbol{\Theta}(\hat{\mathcal{X}}, \mathcal{X})$ be the maximum angle between $\hat{\mathcal{X}}$ and $\mathcal{X}$ that are subspaces spanned by $\hat{\boldsymbol{X}}$ and $\boldsymbol{X}$, respectively. Then by the Davis-Kahan's second $\sin\theta$ theorem[35], we have

$$\|\sin\boldsymbol{\Theta}(\hat{\mathcal{X}}, \mathcal{X})\|_{\mathrm{F}} \leqslant \frac{\|\boldsymbol{E}\|_F}{\lambda_{N+1}-\lambda_N-\|\boldsymbol{E}\|_2}.$$

Note that (2) implies $(\lambda_{N+1}-\lambda_N)/\|\boldsymbol{A}\|_{\mathrm{F}} \gg u$ and thus $\lambda_{N+1}-\lambda_N \gg \|\boldsymbol{E}\|_{\mathrm{F}} \geqslant \|\boldsymbol{E}\|_2$. Then using the relation

$$\|\sin\Theta(\hat{\mathcal{X}}, \mathcal{X})\|_{\mathrm{F}} = \frac{1}{\sqrt{2}}\|\hat{\boldsymbol{X}}\hat{\boldsymbol{X}}^{\mathrm{T}} - \boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\|_{\mathrm{F}},$$

(see Subsection 4.3 of [35]) and (3), we obtain

$$\begin{aligned}\|\rho(\hat{\boldsymbol{X}}) - \rho(\boldsymbol{X})\|_2 &\leqslant \|\rho(\hat{\boldsymbol{X}}) - \rho(\boldsymbol{X})\|_{\mathrm{F}}\\ &\leqslant \|\hat{\boldsymbol{X}}\hat{\boldsymbol{X}}^{\mathrm{T}} - \boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\|_{\mathrm{F}}\\ &\leqslant \frac{\sqrt{2}\|\boldsymbol{E}\|_{\mathrm{F}}}{\lambda_{N+1}-\lambda_N-\|\boldsymbol{E}\|_2}\\ &= O\Big(\frac{\|\boldsymbol{A}\|_{\mathrm{F}}u}{\lambda_{N+1}-\lambda_N}\Big).\end{aligned}$$

Therefore, condition (2) leads to $\|\rho(\hat{\boldsymbol{X}})-\rho(\boldsymbol{X})\|_2 \ll \varepsilon$, which implies that $\rho(\hat{\boldsymbol{X}})$ has the same accuracy as $\hat{\rho}$. □

Note that the roundoff unit u describes the limit of precision for storing and computing corresponding to the specific floating-point format in a machine, because u measures the relative error of rounding that means $|\mathrm{fl}(x)-x|/|x| \leqslant u$ for any nonzero real number x (see more details in [34]). For insulator and semiconductor systems, the band gap between the occupied orbitals and undesired ones is relatively large, which implies a relative big value of $\lambda_{N+1}-\lambda_N$. Since the roundoff unit of single precision is $u = 2^{-24} \approx 5.96\times10^{-8}$ and ε is not very small (seldom smaller than 10^{-3}), the condition (2) is almost always satisfied. Numerical results will show that our mixed precision variant of the AB_CG algorithm does take less time to converge than the original one while maintaining the same accuracy.

Remark 1. *By the first Hohenberg-Kohn theorem*[36], *the Hamiltonian (at most up to a constant) is determined uniquely by the ground state charge density ρ, and thus the total energy E_{tol} depends only on ρ. Note that the mapping relation from ρ to E_{tol} is nonlinear and very complex, not to mention that the exact analytical form of the exchange-correlation energy term is unknown. This makes it almost impossible to give a rigorous analysis of the accuracy of computed total energy. However, a large amount of practical computation experiences in this field confirm that an accurately computed ρ can lead to an accurately computed E_{tol}.*

3.3 System Innovations

We have also made the following four optimizations for the heterogeneous DCU system.

3.3.1 *Coarse-Grained Parallelism*

A two-layer parallel architecture (as illustrated in Fig.3) is proposed in the original LS3DF algorithm, where task parallelism is in the first layer and data parallelism is in the second layer. The first layer divides the entire system into multiple fragments,

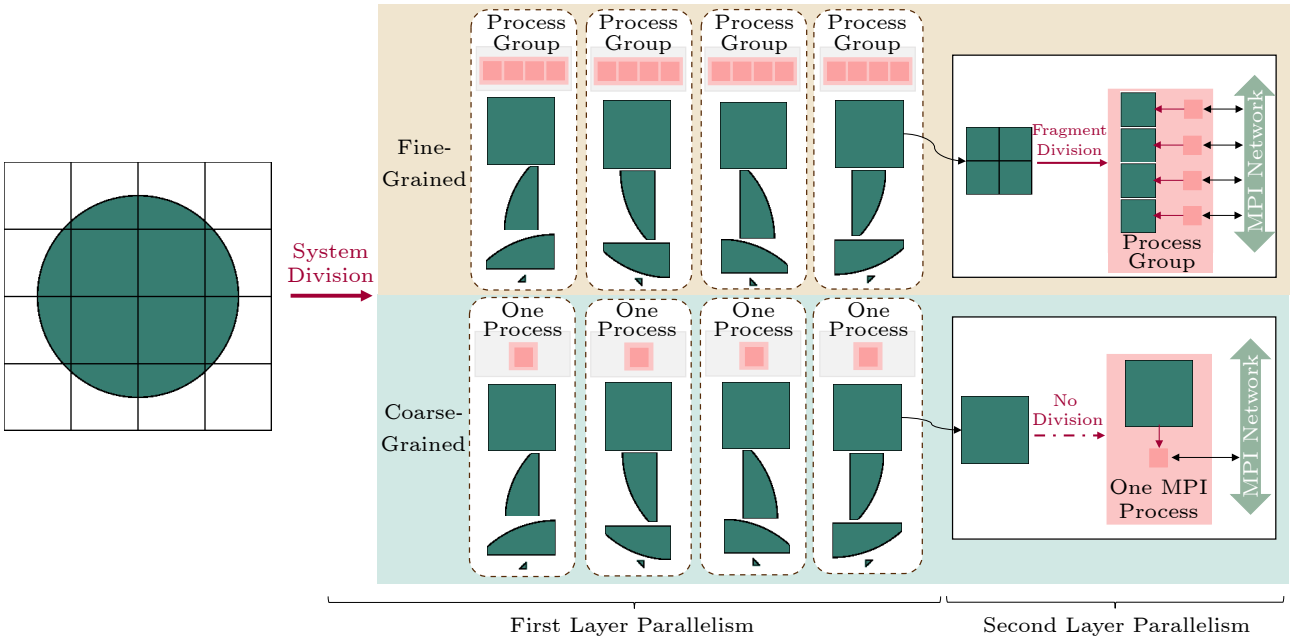


Fig.3. Two-layer parallelism of LS3DF. It illustrates the partitioning method of the entire system based on different granularities. The upper row represents the original fine-grained parallel, and the under row represents the optimized coarse-grained parallel. The green blocks represent tasks that require calculations following the division of the system. The pink squares represent the MPI processes. The red arrows indicate the mapping between tasks and processes.

whose computational task is independent of each other. In the second layer of parallelism, each fragment is further divided into small pieces based on the number of processes assigned to it. Only local information within each piece is distributed to its process. The task parallelism and the data parallelism compose a fine-grained parallel strategy.

Fine-grained parallelism increases data transfer overhead while speeding up computations. The division strategy requires multiple processes to handle one fragment simultaneously and, as a result, a large amount of data must be communicated between these processes. These communications are conducted through the message passing interface (MPI). Furthermore, memory copy operations between accelerator DCUs and CPUs on a heterogeneous platform bring about additional transfer overhead. The MPI communications and the memory copy operations occur frequently during the computations. The data-hungry caused by MPI communications and memory copy operations will stall the DCU pipeline. The transfer overhead becomes a bottleneck given the rapid increase in the computation performance of accelerators, which will be wasted if fine-grained parallelism is still used.

Coarse-grained parallelism is proposed to reduce the data transfer overhead caused by fine-grained parallelism. Essentially, it removes the second layer of a two-layer parallel architecture. Specifically, fragments in the second layer parallelism will no longer be divided among multiple processes, but instead will be calculated by a single process. Before coarse-grained optimization, the wave function is scheduled to be in many processes, and each process owns part of the wave function. Therefore, after each step of the AB_CG calculation, MPI_allreduce or MPI_alltoallv operations are needed for the wave function. By our count, 21 MPI_allreduce operations and four MPI_alltoallv operations are needed in the whole AB_CG computation. Moreover, most of the computations are performed on the DCUs, and the DCUs cannot communicate with each other directly, and thus all data needs to be copied to the CPU first before communication. According to the statistics, 43 data copies are needed for the whole computation. By using this approach, the large amount of data transfer necessary for the original fragment computation is not required. The whole 25 times MPI communications are no longer required during the computation. And the data transfer only needs three times, copying the wave function and Hamiltonian matrix from the CPU to the DCU before the calculation starts, and copying the calculation result from the DCU back to the CPU after the calculation is finished. In this way, a large number of bubbles that occur in the DCU pipeline due to waiting for data transfer disappear; thus the

computation efficiency is enhanced.

3.3.2 Multi-Stream 3D FFT

The computational complexity of the AB_CG algorithm decreases significantly with the coarse-grained optimization, but it still occupies 61.3% of the overall SCF iteration time. The *Hpsi* functions in line 2 and line 12 of Algorithm 1 are the most time-consuming among the full AB_CG algorithm, taking up to 83.9% of the computation time. The *Hpsi* function multiplies a group of band wave functions $\boldsymbol{\Psi}$ by Hamiltonian operator $\boldsymbol{H}$. While the full matrix $\boldsymbol{H}$ requires prohibitively large amounts of memory, it is commonly decomposed into a kinetic energy operator $-1/2\nabla^2$, a local potential $V(\boldsymbol{r})$, and a non-local pseudo-potential projector $\sum_l |\phi_l><\phi_l|$. Hence,

$$\boldsymbol{H\Psi}_i = \left[-\frac{1}{2}\nabla^2 + V(\boldsymbol{r}) + \sum_{l=1}^{N} |\phi_l><\phi_l|\right]\boldsymbol{\Psi}_i. \quad (4)$$

As the most time-consuming component of the Hamiltonian operator $\boldsymbol{H}$, the local pseudo-potential projector consumes 74% execution time of the whole *Hpsi* function.

The multiplication of local pseudo-potential projector and wave functions $\boldsymbol{\Psi}$ breaks into five steps as shown in Fig.4: 1) padding the sphere wave functions into cubes in the frequency space; 2) transforming the padded cubes into the real space by inverse Fourier transform; 3) multiplying the wave functions with corresponding residual potential terms in the real space; 4) transforming the cubes back to the frequency space by forwarding Fourier transform; 5) mapping the cubes back to spheres. These five steps are all memory-intensive operations and are unable to use all compute units in DCU cards. The aforementioned operations are repeated several times with each wave function, which increases the computational overhead furthermore.

We propose a multi-stream 3D Fourier transform approach to address the above issues. A DCU stream is a sequence of tasks that execute in issue order, and DCU tasks from different streams can be interleaved. Therefore, three new streams are created on each DCU card. The multiplication tasks are assigned to streams in a cyclic order and computed in parallel according to the characteristics of DCUs. When the kernel functions of multiplication tasks on one stream cannot occupy all compute units, the kernel functions on other streams will be launched on the remaining compute units to increase DCU utilization and speed up the multiplications. Fig.4 shows that after multi-stream 3D FFT optimization, the kernel functions in each stream can be computed on the DCU simultaneously.

3.3.3 Accelerating Other Calculations with DCUs

We accelerate the following three calculations with DCUs.

- The first is the non-local pseudopotential projector in the *Hpsi* function. The original implementation requires an All-Reduce MPI communication due to fine-grained parallelism and causes extra data transfer between CPUs and DCUs. With our coarse-grained parallelism, the remaining calculations of the non-local projector are implemented on DCUs to avoid unnecessary data transfers.
- The second is the Rho_cal algorithm of SCF iteration, which is the main contributor to computation time after the AB_CG algorithm optimization.
- The third is the force analysis for each atom. LS3DF is commonly adopted for molecular dynamics

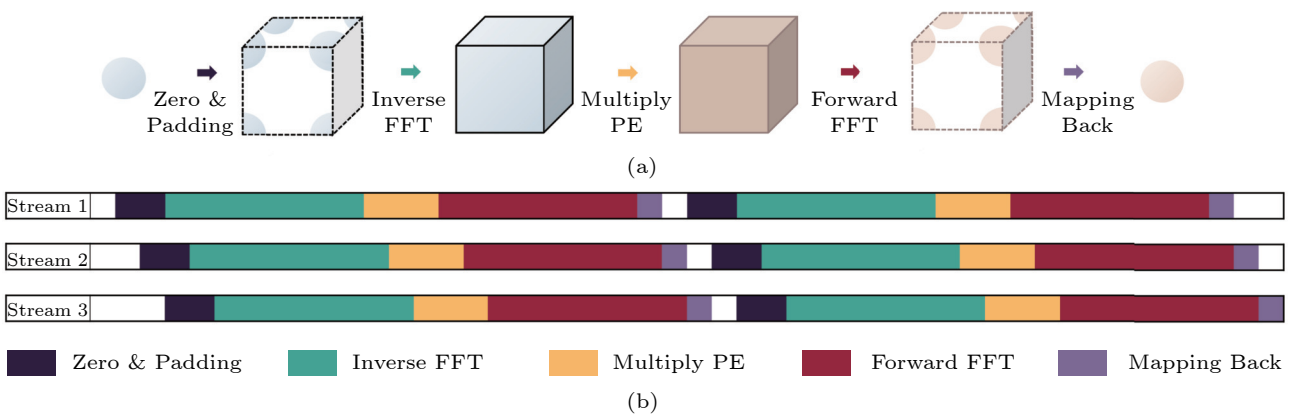


Fig.4. Multi-stream 3D FFT approach. (a) Five-step process of the local pseudo-potential projector multiplied by the wave function in the actual calculation. (b) Pipeline for each stream on DCUs following the adoption of the multi-stream 3D FFT method. The different colored bars represent the corresponding calculation in the upper steps.

simulation and the force analysis is computation-intensive in a large system.

3.3.4 Kernel Fusion and Redundancy Elimination

Most complex computations are migrated to DCUs with the above optimization strategies. Some of their kernels suffer from efficiency issues. We provide two tricks for efficiency improvements.

Kernel fusion combines multiple simple kernels into one complex kernel. The original implementation includes a large number of simple kernels like matrices multiplied with a scalar, which leads to extra kernel launch expenses and decreases DCU utilization. We fuse these sequential simple kernels into one kernel and reduce computation overhead.

Redundancy elimination omits redundant computation according to matrix characteristics. The Cholesky decomposition is required for the AB_CG algorithm, which involves two kernel functions, one multiplying a matrix with a scalar, and the other subtracting the upper triangle for the following calculations. The scalar multiplication for the lower triangle in the first kernel is redundant, and thus we provide a new kernel to subtract the upper triangle first and then multiply it by the scalar.

4 Physical Systems

All numerical tests are performed on a bulk silicon system ranging from 8 000 to 10 million atoms. The plane wave energy cutoff is set to 20 Ryd (Rydberg), and norm-conserving pseudo-potential is used. Four nonlocal projectors are evaluated in the G-space, which can be more accurate compared with the real-space implementation. On average, it takes about 13 SCF steps to converge to 1.0×10^{-5} for the relative error of charge density. In the 10 648 000 atom system, the total number of electrons is 42 592 000. The total number of real-space grids for charge density is $2\,420^3$. For a typical fragment of 219 atoms, the G-space grid is 4.86×10^4, and the corresponding real space grid is 66^3. Note that AB_CG is the most computationally intensive part of the SCF calculation. The total number of floating point operations (FLOPs) for the AB_CG per SCF is 5.18×10^{18}, which is collected via counting the effective FLOPs. Note that silicon is the most important material in the semiconductor industry, and the 10 million atoms system can be used in modeling next-generation semiconductor devices.

5 Machine Configuration

The DCU cluster at Xi'an National Supercomputing Center, which is capable of 164.16 PFLOPS peak performance, is used as the test platform. This system consists of 3 800 computing nodes, each of which is equipped with a 32-core CPU and four DCU cards. The CPU is a Hygon 32-core processor running at 2 GHz, with 8×16 DDR4 system memory. The DCUs refer to deep computing units, a class of co-processors designed by Hygon based on the general-purpose graphics processing unit (GPGPU) architecture, suitable for computationally intensive and computing acceleration fields. The DCU cards with 16 GB of global memory are installed in the computing node. With each card, 21.6 TFLOPS can be provided for single-precision operation and 10.8 TFLOPS for double-precision operation. And all mixed-precision calculations are performed in IEEE 754 compliant FP32 (vs FP64). Four DCU cards are connected to 32 CPU cores by PCI-Express bus lanes. A 200 Gb Infiniband connection is provided by Mellanox to the computing nodes. The development of the entire software was carried out on the ORISE supercomputer, which has a similar architectural system to the Xi'an National Supercomputing Center.

An MPI[37]+OpenMP+ROCm[②] programming model is incorporated into the optimized LS3DF. A computing node is launched with four MPI tasks, with each MPI task corresponding to eight CPU cores and one DCU card. Various compilers are used, including the Fortran compiler gfortran, the C++ compiler g++, and the MPI wrapper compilers mpif90 and mpicxx. In addition, the implementation requires the FFT[③], BLAS[38], LAPACK[39], ScaLAPACK[40] (only for CPU version), and MAGMA[41] libraries.

6 Numerical Results

As part of this section, we first illustrate the convergence of LS3DF after the algorithmic improvements. Then we illustrate the speedup ratios of LS3DF obtained from each optimization strategy on a small-scale system. In addition, the tests are sequentially extended to large-scale systems to illustrate their scalability.

[②] https://github.com/ROCm/ROCm, July 2023.
[③] https://github.com/ROCm/hipFFT, July 2023.

6.1 Convergence and Accuracy

6.1.1 *Improved AB_CG Algorithm*

Regarding the improved AB_CG algorithm, we test the convergence performance on a system of 8 000 silicon atoms. The total SCF iteration is set to 13 in this test, and the charge density is calculated for the whole system at each iteration.

For the original and improved AB_CG, Fig.5 depicts the residual norms for the 1st, 5th, 9th, and 13th SCF iteration with the form $\sum_{i=1}^{N} \|H^{(k)}\Psi_i^{(k)} - \varepsilon_i^{(k)}\Psi_i^{(k)}\|_2^2$, respectively. We can find that the residual norms of the improved AB_CG are always smaller than that of the original one, and thus the improved algorithm can converge more quickly, which is due to the choice of optimal angle. Besides, we can find that the improved AB_CG can save at least one step at each SCF iteration while can obtain final results with the same accuracy in practical computations.

6.1.2 *Mixed Precision Computations*

For comparisons between double precision and mixed precision implementations of AB_CG, we run 16-step SCF iterations for the same system as the above. The computation of $\Theta_i = HP_i$ is performed using double precision and single precision, respectively, while the other parts of the algorithm are all performed by using double precision.

Fig.6 depicts the relative error of charge density between the two successive SCF iterations for the two implementations. It can be obviously observed that the two errors are the same for all iterations, which confirms that single precision computation does not affect the convergence and accuracy of the SCF results.

We also compare the accuracy of the converged total energy obtained by double precision and mixed precision computations. For the system of 8 000 silicon atoms, after the SCF iterations of 16 rounds, the difference of total energy obtained by the two computations is less than 10^{-14} Ryd.

6.2 Small System

Through small-scale nodes, we test the benefits of algorithmic and system innovations. The 8 000 silicon atoms system is utilized, using three computing nodes per test. The number of SCF iterations is set to 14 in

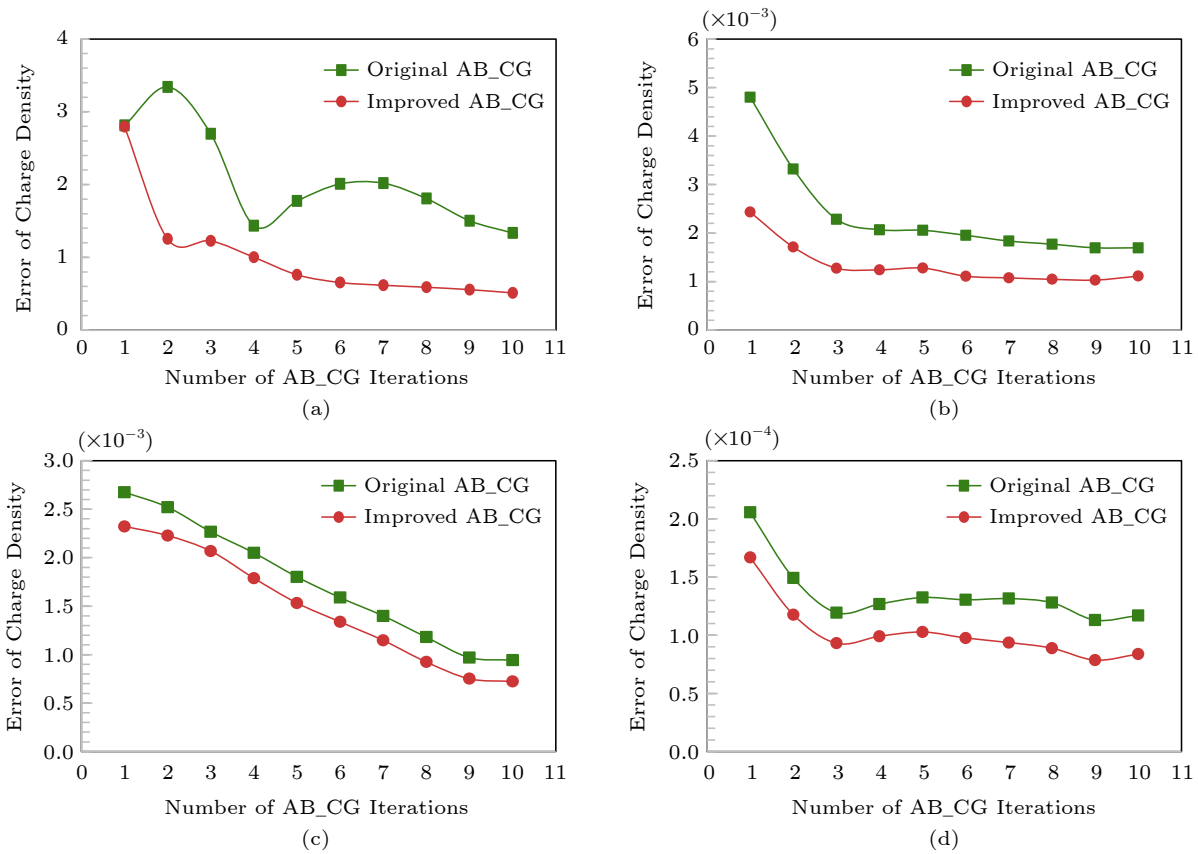


Fig.5. Convergence analysis of the improved AB_CG algorithm. (a) 1st, (b) 5th, (c) 9th, (d) 13th SCF iteration.

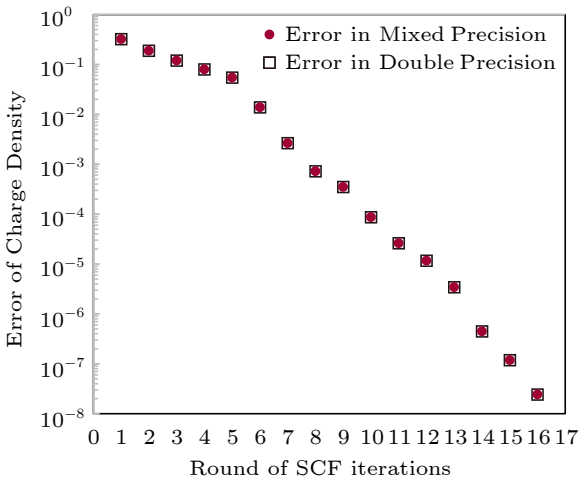


Fig.6. Relative error of charge density between the two successive SCF iterations for double precision and mixed precision implementations.

each test, and the number of AB_CG iterations is set to 4 in each SCF iteration to ensure the same amount of computing rounds is performed throughout all tests. The speedup of the time-to-solution for the LS3DF program on a step-by-step optimization is shown in Fig.7.

6.2.1 *Coarse-Grained Parallelism*

After adopting coarse-grained parallel innovation, massive MPI communication as well as data copying between CPUs and DCUs for the AB_CG algorithm is avoided. Based on the analysis of all AB_CG computation time, we find that the time of this part is reduced by 44.9% as a result of the optimization. Additionally, LS3DF spends most of its time in AB_CG computations; thus it is 1.5 times faster than the baseline after this optimization.

6.2.2 *Multi-Stream 3D FFT*

The AB_CG algorithm remains to be the most time-consuming part after coarse-grained parallel optimization; therefore all computations in this subsection are analyzed. Our analysis of the calculation time for each part of AB_CG has revealed that the FFT calculation is the most time-consuming, accounting for 36.9% of the overall LS3DF time. For this reason, we propose an optimization method based on multi-stream 3D FFT to mend this memory-bond calculation. Using this method, we have tried to generate two, three, and four streams, respectively, and we find that the time required for AB_CG is reduced by 9.2%, 17.4%, and 17.2%. It can be seen that the three streams case gives the best result. Because three streams can utilize all computing resources to their full potential, increasing the number of streams does not necessarily result in greater benefits. Instead, there is some overhead associated with stream creation and destruction. Based on a three-stream selection, multi-stream 3D FFT results in a speedup of 1.8 times over the baseline.

6.2.3 *Accelerating Other Calculations with DCUs*

Recent years have seen an increase in the gap between the computing capabilities of CPUs and DCUs. Eight cores have computing capabilities of 32 GFLOPS, corresponding to a DCU card that has 10.8 TFLOPS. There is a 337.5-fold difference between the two. In this regard, we have also transferred the remaining calculation to DCUs in the AB_CG method. As a result, the time proportion of the AB_CG part

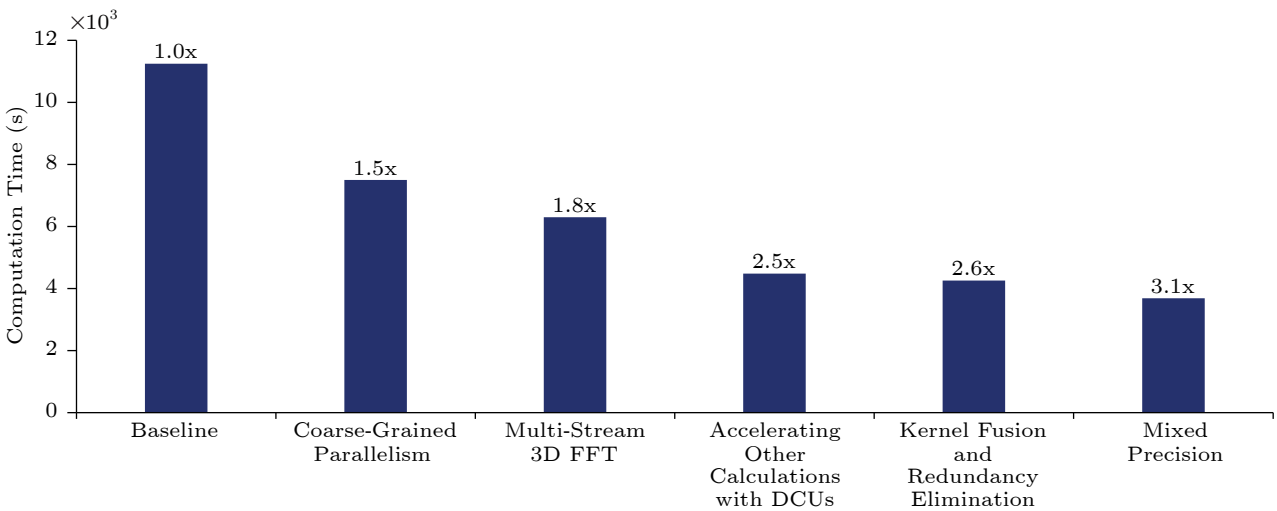


Fig.7. Step-by-step optimization and their corresponding speedup for the 8 000 silicon atoms system.

has been further reduced from 60.0% to 54.2%. Furthermore, as the calculation time of the AB_CG part continues to decrease, we transfer the time-consuming density and force calculations to DCUs. In particular, the total time taken to calculate density is accelerated by an amount of 8.2, and the calculation of force is accelerated by an amount of 14.8. In comparison to the baseline, the optimizations up to this point have increased the speed by 2.5x.

6.2.4 *Kernel Fusion and Redundancy Elimination*

Prior optimizations essentially shifted all calculations from CPUs to DCUs. The analysis of the kernel functions implemented on each DCU reveals that, among the 43 kernel functions that we implemented and the kernel functions that are called by the BLAS library, many are capable of performing fusion operations. The fused kernel function is reduced to 32. After fusion, some redundant operations can be eliminated. The total speedup is 2.6 compared with the baseline.

6.2.5 *Mixed Precision*

After the rest of the calculations have also been performed on the DCUs, the time proportion of AB_CG increases to 92.3%. At this point, we use mixed precision to reduce the time of AB_CG even further. Because this is an iterative algorithm, we have changed the *Hpsi* function, which accounts for 86.85% of the time, from double precision to single precision. Overall, the speedup factor reaches 3.1.

6.3 Scaling Towards Large-Scale Systems

6.3.1 *Strong Scaling*

On a 1-million silicon atoms system, we perform strong scalability tests for the optimized LS3DF method. The scaling behavior ranges from 375 to 3 000 computing nodes. Throughout all tests, the number of SCF iterations is set to 5 and the time-to-solution is recorded.

Since our method is linear scaling, it is originally designed with large-scale systems. As shown in Fig.8, the optimized LS3DF can guarantee a good strong scaling for large-scale systems. For computing the 1 million silicon atoms system, the majority of the time-consuming part is fragment computations. In this system, there are $50 \times 50 \times 50$ fragments and each process must deal with multiple fragments. As the number of nodes increases, the fragments allocated to each process decrease proportionally, thus reducing the total calculation time. Furthermore, it is evident in Fig.8 that when the number of nodes reaches 3 000, there is a gap with the perfect linear scaling. The gap is caused by the global communication of integrating computational results of all fragments.

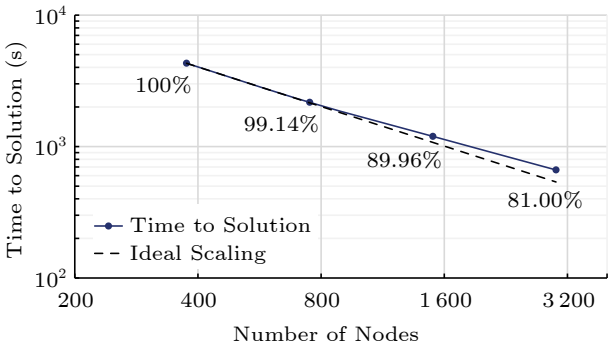


Fig.8. Strong scaling of 1 million silicon atoms system (five SCF iterations).

6.3.2 *Weak Scaling*

The weak scaling of the optimized AB_CG method is measured in terms of the system size and FLOPS for silicon systems (Fig.9). Here, we increase the number of atoms from 8 000 to 10 million while increasing the number of nodes from 3 to 3 800 proportionally. The result illustrates that the AB_CG method reaches perfect weak scaling as the system size increases. As a result, it can achieve 34.8 PFLOPS at the scale of 10 million atoms (21.2% of the theoretical peak). Here, the "theoretical performance" means the theoretical peak of 3 800 computing nodes. Compared with the DGDFT method, the system size we can calculate extends by 10. Due to the perfect linear scaling, we foresee that our optimized LS3DF method can compute larger physical systems on future supercomputers.

7 Conclusions

In this paper, we demonstrated that our optimized LS3DF can be used to simulate electronic devices of more than 10-million atoms from first-principle calculations. Test results showed that our optimized LS3DF is 3.1 times more efficient than the original heterogeneous version, and the corresponding

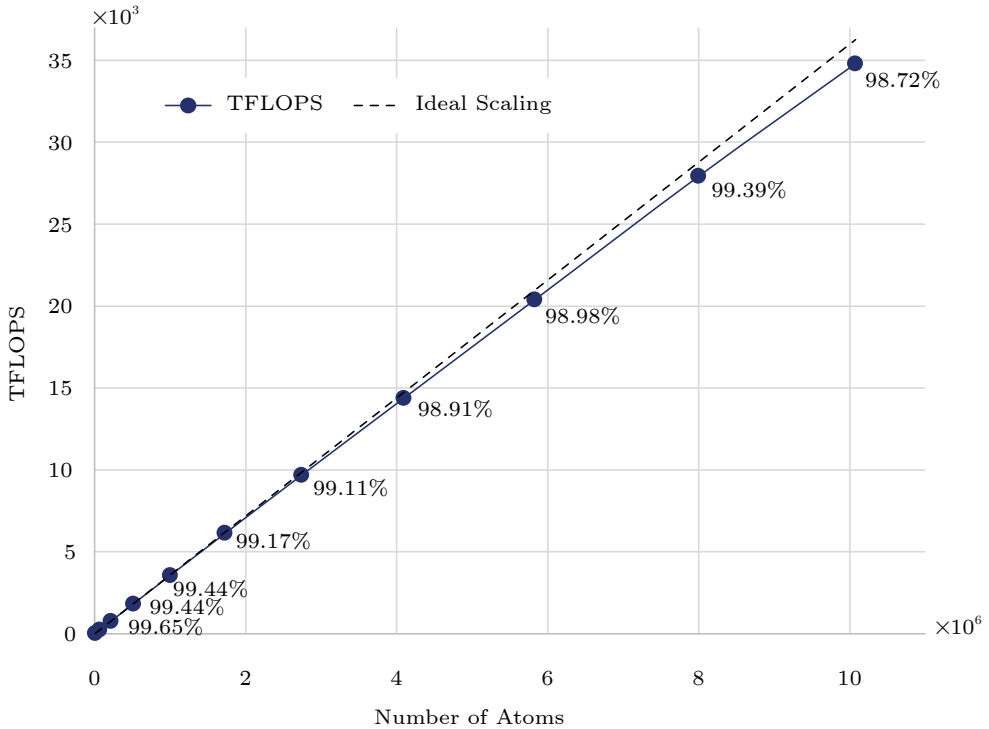


Fig.9. Weak scaling of the AB_CG method with respect to the number of atoms.

parallel efficiency reaches 81% when scaling from 400 to 3 200 computing nodes. The weak scaling shows that our code can reach nearly perfect scaling and 21.2% of peak (34.8 PFLOPS) for a 10 million-atom system. Such system scaling makes it possible to model electronic devices such as FETs on domestic supercomputers. Although our optimizations are implemented on the Sugon platform, the optimization strategy shown in this paper can also be applied to other heterogeneous architectures such as Sunway and NIVIDA GPU.

In our optimized LS3DF code, fragment calculation such as the evaluation of AB_CG still takes 92.4% of the total time. Thus we plan in the future to adopt more lower-precision computations in the SCF iteration to achieve better performance. Furthermore, the most time consuming parts of all calculations are matrix multiplication and Fast Fourier transform. Therefore, to further improve the efficiency, these two kernel functions need to be optimized. Moreover, the current implementation of LS3DF is limited to solving general problems. In our future plans, the load-balance problem can be one direction for optimization, and we will choose dynamical scheduling methods for more complex problems.

Conflict of Interest The authors declare that they have no conflict of interest.

References

[1] Naveh Y, Likharev K K. Shrinking limits of silicon MOSFETs: Numerical study of 10 nm scale devices. *Superlattices and Microstructures*, 2000, 27(2/3): 111–123. DOI: 10.1006/spmi.1999.0807.

[2] Ravaioli U. Quantum phenomena in semiconductor nanostructures. In *Encyclopedia of Complexity and Systems Science*, Meyers R A (ed.), Springer, 2009, pp.7400–7422. DOI: 10.1007/978-0-387-30440-3_439.

[3] Kohn W, Sham L J. Self-consistent equations including exchange and correlation effects. *Physical Review*, 1965, 140(4A): A1133–A1138. DOI: 10.1103/PhysRev.140.A1133.

[4] Payne M C, Teter M P, Allan D C, Arias T A, Joannopoulos J D. Iterative minimization techniques for *ab initio* total-energy calculations: Molecular dynamics and conjugate gradients. *Reviews of Modern Physics*, 1992, 64(4): 1045–1097. DOI: 10.1103/RevMod-Phys.64.1045.

[5] Kresse G, Furthmüller J. Efficient iterative schemes for *ab initio* total-energy calculations using a plane-wave basis set. *Physical Review B*, 1996, 54(16): 11169–11186. DOI: 10.1103/PhysRevB.54.11169.

[6] Tsuchida E, Tsukada M. Electronic-structure calculations based on the finite-element method. *Physical Review B*, 1995, 52(8): 5573–5578. DOI: 10.1103/PhysRevB.52.5573.

[7] Suryanarayana P, Gavini V, Blesgen T, Bhattacharya K, Ortiz M. Non-periodic finite-element formulation of Kohn–Sham density functional theory. *Journal of the Mechanics and Physics of Solids*, 2010, 58(2): 256–280. DOI:

10.1016/j.jmps.2009.10.002.

[8] Bao G, Hu G H, Liu D. An *h*-adaptive finite element solver for the calculations of the electronic structures. *Journal of Computational Physics*, 2012, 231(14): 4967–4979. DOI: 10.1016/j.jcp.2012.04.002.

[9] Chen H J, Dai X Y, Gong X G, He L H, Zhou A H. Adaptive finite element approximations for Kohn–Sham models. *Multiscale Modeling & Simulation*, 2014, 12(4): 1828–1869. DOI: 10.1137/130916096.

[10] Das S, Motamarri P, Gavini V, Turcksin B, Li Y W, Leback B. Fast, scalable and accurate finite-element based *ab initio* calculations using mixed precision computing: 46 PFLOPS simulation of a metallic dislocation system. In *Proc. the 2019 International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2019, Article No. 2. DOI: 10.1145/3295500.3357157.

[11] Gygi F, Draeger E W, Schulz M, de Supinski B R, Gunnels J A, Austel V, Sexton J C, Franchetti F, Kral S, Ueberhuber C W, Lorenz J. Large-scale electronic structure calculations of high-Z metals on the BlueGene/L platform. In *Proc. the 2006 ACM/IEEE Conference on Supercomputing*, Nov. 2006, Article No. 45. DOI: 10.1145/1188455.1188502.

[12] Wang L W, Lee B, Shan H Z, Zhao Z J, Meza J, Strohmaier E, Bailey D H. Linearly scaling 3D fragment method for large-scale electronic structure calculations. In *Proc. the 2008 ACM/IEEE Conference on Supercomputing*, Nov. 2008. DOI: 10.1109/SC.2008.5218327.

[13] Hasegawa Y, Iwata J, Tsuji M, Takahashi D, Oshiyama A, Minami K, Boku T, Shoji F, Uno A, Kurokawa M, Inoue H, Miyoshi I, Yokokawa M. First-principles calculations of electron states of a silicon nanowire with 100 000 atoms on the K computer. In *Proc. the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2008, Article No. 1. DOI: 10.1145/2063384.2063386.

[14] Nakata A, Baker J S, Mujahed S Y, Poulton J T L, Arapan S, Lin J B, Raza Z, Yadav S, Truflandier L, Miyazaki T, Bowler D R. Large scale and linear scaling DFT with the CONQUEST code. *The Journal of Chemical Physics*, 2020, 152(16): 164112. DOI: 10.1063/5.0005074.

[15] Kühne T D, Iannuzzi M, Del Ben M, Rybkin V V, Seewald P, Stein F, Laino T, Khaliullin R Z, Schütt O, Schiffmann F, Golze D, Wilhelm J, Chulkov S, Bani-Hashemian M H, Weber V, Borštnik U, Taillefumier M, Jakobovits A S, Lazzaro A, Pabst H, Müller T, Schade R, Guidon M, Andermatt S, Holmberg N, Schenter G K, Hehn A, Bussy A, Belleflamme F, Tabacchi G, Glöß A, Lass M, Bethune I, Mundy C J, Plessl C, Watkins M, Vandevondele J, Krack M, Hutter J. CP2K: An electronic structure and molecular dynamics software package-quickstep: Efficient and accurate electronic structure calculations. *The Journal of Chemical Physics*, 2020, 152(19): 194103. DOI: 10.1063/5.0007045.

[16] Shang H H, Li F, Zhang Y Q, Zhang L B, Fu Y, Gao Y X, Wu Y J, Duan X H, Lin R F, Liu X, Liu Y, Chen D X. Extreme-scale *ab initio* quantum Raman spectra simulations on the leadership HPC system in China. In *Proc. the 2021 International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2021, Article No. 6. DOI: 10.1145/3458817.3487402.

[17] Hu W, Qin X M, Jiang Q C, Chen J S, An H, Jia W L, Li F, Liu X, Chen D X, Liu F F, Zhao Y W, Yang J L. High performance computing of DGDFT for tens of thousands of atoms using millions of cores on Sunway TaihuLight. *Science Bulletin*, 2021, 66(2): 111–119. DOI: 10.1016/j.scib.2020.06.025.

[18] Schade R, Kenter T, Elgabarty H, Lass M, Schütt O, Lazzaro A, Pabst H, Mohr S, Hutter J, Kühne T D, Plessl C. Towards electronic structure-based *ab-initio* molecular dynamics simulations with hundreds of millions of atoms. *Parallel Computing*, 2022, 111: 102920. DOI: 10.1016/j.parco.2022.102920.

[19] Hu W, An H, Guo Z Q, Jiang Q C, Qin X M, Chen J S, Jia W L, Yang C, Luo Z L, Li J L, Wu W T, Tan G M, Jia D N, Lu Q L, Liu F F, Tian M, Li F, Huang Y Q, Wang L Y, Liu S, Yang J L. 2.5 million-atom *ab initio* electronic-structure simulation of complex metallic heterostructures with DGDFT. In *Proc. the 2022 International Conference on High Performance Computing, Networking, Storage and Analysis*, Nov. 2022, Article No. 5. DOI: 10.1109/SC41404.2022.00010.

[20] Goedecker S. Linear scaling electronic structure methods. *Reviews of Modern Physics*, 1999, 71(4): 1085–1123. DOI: 10.1103/RevModPhys.71.1085.

[21] Lin L, Lu J F, Car R, E W N. Multipole representation of the Fermi operator with application to the electronic structure analysis of metallic systems. *Physical Review B*, 2009, 79(11): 115133. DOI: 10.1103/PhysRevB.79.115133.

[22] Bowler D R, Miyazaki T. $O(N)$ methods in electronic structure calculations. *Reports on Progress in Physics*, 2012, 75(3): 036503. DOI: 10.1088/0034-4885/75/3/036503.

[23] Wang L W, Zhao Z J, Meza J. Linear-scaling three-dimensional fragment method for large-scale electronic structure calculations. *Physical Review B*, 2008, 77(16): 165113. DOI: 10.1103/PhysRevB.77.165113.

[24] Ye M, Jiang X W, Li S S, Wang L W. Large-scale ab initio quantum transport simulation of nanosized copper interconnects: The effects of defects and quantum interferences. In *Proc. the 2019 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2019, Article No. 24. DOI: 10.1109/IEDM19573.2019.8993549.

[25] Wang L W, Jia W L, Cao Z Y, Wang L, Chi X B, Gao W G. GPU speedup of the plane wave pseudopotential density functional theory calculations. In *APS March Meeting Abstracts*, Feb. 27–March 2, 2012, Abstract ID T7.008.

[26] Tomo S, Langou J, Dongarra J, Canning A, Wang L W. Conjugate-gradient eigenvalue solvers in computing elec-

tronic properties of nanostructure architectures. *International Journal of Computational Science and Engineering*, 2006, 2(3/4): 205–212. DOI: 10.1504/IJCSE.2006.012774.

[27] Kohn W. Density functional and density matrix method scaling linearly with the number of atoms. *Physical Review Letters*, 1996, 76(17): 3168–3171. DOI: 10.1103/PhysRevLett.76.3168.

[28] Auckenthaler T, Blum V, Bungartz H J, Huckle T, Johanni R, Krämer L, Lang B, Lederer H, Willems P R. Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations. *Parallel Computing*, 2011, 37(12): 783–794. DOI: 10.1016/j.parco.2011.05.002.

[29] Yang C, Meza J C, Wang L W. A trust region direct constrained minimization algorithm for the Kohn–Sham equation. *SIAM Journal on Scientific Computing*, 2007, 29(5): 1854–1875. DOI: 10.1137/060661442.

[30] Vecharynski E, Yang C, Pask J E. A projected preconditioned conjugate gradient algorithm for computing many extreme eigenpairs of a Hermitian matrix. *Journal of Computational Physics*, 2015, 290: 73–89. DOI: 10.1016/j.jcp.2015.02.030.

[31] Knyazev A V. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM Journal on Scientific Computing*, 2001, 23(2): 517–541. DOI: 10.1137/S1064827500366124.

[32] Jia W L, Cao Z Y, Wang L, Fu J Y, Chi X B, Gao W G, Wang L W. The analysis of a plane wave pseudopotential density functional theory code on a GPU machine. *Computer Physics Communications*, 2013, 184(1): 9–18. DOI: 10.1016/j.cpc.2012.08.002.

[33] Fattebert J L, Osei-Kuffuor D, Draeger E W, Ogitsu T, Krauss W D. Modeling dilute solutions using first-principles molecular dynamics: Computing more than a million atoms with over a million cores. In *Proc. the 2016 International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2016, pp.12–22. DOI: 10.1109/SC.2016.88.

[34] Higham N J. Accuracy and Stability of Numerical Algorithms. SIAM, 2002.

[35] Sun J G. Matrix Perturbation Analysis (2nd edition). Science Press, 2001. (in Chinese)

[36] Hohenberg P, Kohn W. Inhomogeneous electron gas. *Physical Review*, 1964, 136(3B): B864–B871. DOI: 10.1103/PhysRev.136.B864.

[37] Gabriel E, Fagg G, Bosilca G *et al.* Open MPI: Goals, concept, design of a next generation MPI implementation. In *Proc. the 11th European PVM/MPI Users' Group Meeting*, Sept. 2004, pp.97–104. DOI: 10.1109/CLUSTR.2006.311904.

[38] Van Zee F G, van de Geijn R A. BLIS: A framework for rapidly instantiating BLAS functionality. *ACM Trans. Mathematical Software*, 2015, 41(3): Article No. 14. DOI: 10.1145/2764454.

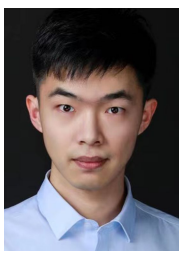
[39] Anderson E, Bai Z, Bischof C, Blackford L S, Demmel J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Sorensen D. LAPACK Users' Guide (3rd edition). Society for Industrial and Applied Mathematics, 1999.

[40] Blackford L S, Choi J, Cleary A, D'Azevedo E, Demmel J, Dhillon I, Dongarra J, Hammarling S, Henry G, Petitet A, Stanley K, Walker D, Whaley R C. ScaLAPACK Users' Guide. Society for Industrial and Applied Mathematics, 1997.

[41] Bosma W, Cannon J, Playoust C. The Magma algebra system I: The user language. *Journal of Symbolic Computation*, 1997, 24(3/4): 235–265. DOI: 10.1006/jsco.1996.0125.

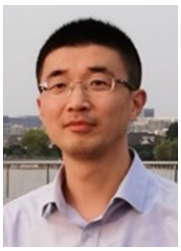
Yu-Jin Yan is currently a Ph.D. candidate in the State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences, and University of Chinese Academy of Sciences, Beijing. She received her B.S. degree in mathematics and applied mathematics from Sichuan University, Chengdu, in 2017. Her current research interests include high-performance computing, massively parallel computing, and first-principles calculation.

Hai-Bo Li received his Ph.D. degree in computational mathematics from Tsinghua University, Beijing, in 2021. He is currently a joint postdoctoral researcher at the Computing System Optimization Laboratory of Huawei Technologies, Beijing, and Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His research interests include numerical linear algebra, computational inverse problems, and machine learning.

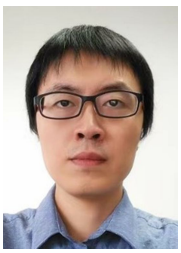


Tong Zhao received his Ph.D. degree in operation research and control theory from Fudan University, Shanghai, in 2021. He is currently a postdoctoral researcher at the State Key Laboratory of Processors, the Institute of Computing Technology, Chinese Achademy of Science, Beijing. His research interests include fundamental theory of artificial intelligence, high-performance computing, and game theory.

Lin-Wang Wang received his Ph.D. degree at Cornell University, Ithaca, in 1991. He worked in Cornell University (1991–1992) and the National Renewable Energy Lab (1992–1995) as a postdoctor and then in Biosym/Molecular Simulations Inc. (1995–1996) and the National Renewable Energy Laboratory (1996–1999) as a staff scientist. From 1999, he has worked in Lawrence Berkeley National Laboratory and is a senior staff scientist. He is currently a chief scientist at the Institute of Semiconductors, Chinese Academy of Sciences, Beijing. His research interests mainly focus on the development of *ab initio* electronic structure calculation methods and the applications of these methods in materials design and discovery.

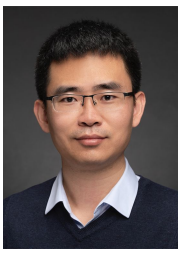


Lin Shi received his B.E. degree in physics from Southeast University, Nanjing, in 2002, and his Ph.D. degree in condensed matter physics from Tsinghua University, Beijing, in 2007. He is currently teaching at the School of Materials Science and Engineering, Yancheng Institute of Technology, Yancheng. His research interests include first-principles calculation and III-V semiconductors.



Tao Liu received his B.E. degree in computer science from Harbin Engineering University, Harbin, in 2002. He is currently a senior engineer at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His research interests include HPC, machine learning, and AI for science applications.

Guang-Ming Tan received his Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in March 2008. He is currently a researcher at the Institute of Computing, Chinese Academy of Sciences, Beijing. His research interests include parallel algorithm design and analysis, parallel programming and optimization, computer architecture, bioinformatics, and big data.



Wei-Le Jia received his joint Ph.D. degree in the Computer Network Information Center, Chinese Academy of Sciences, Beijing, and Lawrence Berkeley National Laboratory, Berkeley, in 2016. He is currently an associate research fellow at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His research interests include high-performance computing, artificial intelligence, and massively parallel computing.

Ning-Hui Sun received his Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in July 1999. He is currently the Academic Director of the Institute of Computing, Chinese Academy of Sciences, Beijing. His research interests include parallel processing architecture, distributed operating systems, performance evaluation, and file systems.