Accelerating All-Electron *Ab initio* Simulation of Raman Spectra for Biological Systems

Honghui Shang*
SKL of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, UCAS
Beijing, China
shanghonghui@ict.ac.cn

Fang Li*
National Supercomputer Center in Wuxi
Wuxi, China
38349735@qq.com

Yunquan Zhang
SKL of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, UCAS
Beijing, China
zyq@ict.ac.cn

Ying Liu
SKL of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
liuying2007@ict.ac.cn

Libo Zhang
National Supercomputer Center in Wuxi
Wuxi, China
zlb03@hotmail.com

Mingchuan Wu
SKL of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, UCAS
Beijing, China
wumingchuan@ict.ac.cn

Yangjun Wu
SKL of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, UCAS
Beijing, China
wuyangjun21@163.com

Di Wei
Tsinghua University
Beijing, China
weid20@mails.tsinghua.edu.cn

Huimin Cui
SKL of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, UCAS
Beijing, China
cuihm@ict.ac.cn

Xin Liu
National Supercomputing Center in Wuxi
Wuxi, China
yyylx@263.net

Fei Wang
Tsinghua University
Beijing, China
f-wang20@mails.tsinghua.edu.cn

Yuxi Ye
Yingxiang Gao
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
sdustyeyuxi@163.com
18810616698@163.com

Shuang Ni
University of Science and Technology of China
Hefei, China
nishuang@163.com

Xin Chen
National Supercomputing Center in Wuxi
Wuxi, China
ischen.xin@foxmail.com

Dexun Chen
Tsinghua University
Beijing, China
adch@263.net

ABSTRACT

Raman spectroscopy provides chemical and compositional information that can serve as a structural fingerprint for various materials. Therefore, simulations of Raman spectra, including both quantum perturbation analyses and ground-state calculations are of significant interest. However, highly accurate full quantum mechanical (QM) simulations of Raman spectra have previously been confined to small systems. For large systems such as biological materials, the computational cost of full QM simulations is extremely high, and their extension to such systems remains challenging. In the work described here, by employing robust new algorithms and advances in implementation for the many-core architectures, we are able to perform fast, accurate, and massively parallel full *ab initio* simulations of the Raman spectra of biological systems with excellent strong and weak scaling, thereby providing a starting point for applying QM approaches to structural studies of such systems.

*Corresponding authors

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SC '21, November 14–19, 2021, St. Louis, MO, USA

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-8442-1/21/11…$15.00
https://doi.org/10.1145/3458817.3476160

KEYWORDS

Quantum mechanics, All-electron, Many-core processor, Scalability, Biological systems

ACM Reference Format:
Honghui Shang, Fang Li, Yunquan Zhang, Ying Liu, Libo Zhang, Mingchuan Wu, Yangjun Wu, Di Wei, Huimin Cui, Xin Liu, Fei Wang, Yuxi Ye, Yingxiang Gao, Shuang Ni, Xin Chen, and Dexun Chen. 2021. Accelerating All-Electron *Ab initio* Simulation of Raman Spectra for Biological Systems. In *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21), November 14–19, 2021, St. Louis, MO, USA.* ACM, Denver, CO, USA, 13 pages. https://doi.org/10.1145/3458817.3476160

1 INTRODUCTION

Raman spectroscopy has proved to be a very powerful analytical tool for the study of biological materials, as it enables effective extraction of biochemical and structural information. Moreover, because Raman spectroscopy does not suffer from interference from water molecules and does not need extensive sample preparation, its range of applications to biological and biomedical studies is rapidly growing.

To fully understand the information provided by Raman spectroscopy, parameter-free ab initio investigation is essential because it can provide a direct link between atomic structure and spectral features, thereby providing an invaluable source of structural information as well as a basis for improving fundamental understanding of physical mechanism between atoms. Raman spectra can be simulated with first-principles approaches using only fundamental theories without fitting parameters from experiments. Such full quantum mechanical (QM)-level simulations of Raman spectra require density-functional perturbation theory (DFPT) in addition to density-functional theory (DFT).

However, the computation of Raman spectra has been traditionally limited to small systems and remains challenging for large systems consisting of up to thousands of atoms. Therefore, a highly efficient acceleration of *ab initio* simulation on modern heterogeneous many-core supercomputers is urgently needed. Here, we optimize the FHI-aims [1] software on the new-generation Sunway heterogeneous many-core supercomputer to address the challenges.

FHI-aims is a massively parallel package for computational molecular and materials science. It contains state-of-the-art DFPT computation modules [2, 3] that can directly link physical properties (phonon dispersions, polarizability, Raman spectra) measured experimentally with the quantum response or perturbation of the system. In Ref.[3], the state-of-the-art formalism of the density-functional perturbation theory has been proposed and implemented in the all-electron FHI-aims code. Just following the theoretical framework proposed in Ref. [3], here in this work, we present a robust new algorithms and implementation advances for FHI-aims to further scale the full QM calculation of Raman spectra to an unprecedented level of 3006 atoms on the new-generation Sunway heterogeneous many-core supercomputer without introducing approximations to the QM beyond those inherent to DFT/DFPT itself.

The major innovations of the current work can be summarized as follows.

- We propose a customized multi-level parallelization scheme for Raman spectra simulation with FHI-aims to achieve high scalability on modern heterogeneous many-core supercomputers.
- We use a series of communication optimizations to reduce the synchronous time.
- We perform direct memory access (DMA) tilings to access the main memory; we also adopt double buffering to overlap computations with memory accesses.
- We use register communication and vector instructions on the Sunway processor to further improve performance.
- We perform a full *ab initio* simulation of a protein containing 3006 atoms and compare it with experimental results. The convergence of this calculation has been achieved by using the self-consistency method.

The optimization methods proposed in this work could be extended to other DFPT code with the same computational characteristics; hence, they will be broadly beneficial to the quantum chemistry, biological, and material science communities.

Based on these developments, highly accurate all-electron *ab initio* calculations could be used to understand the structural information of biological systems. This work provides starting points for applying full QM approaches to the structural studies of biological systems, which allows us to get fundamental understanding about the interactions between atoms within them.

2 BACKGROUND

2.1 Current state of the art

In order to calculate the physical properties (phonon dispersions, polarizability, Raman spectra) theoretically, the quantum perturbation form of the Schrödinger equation needs to be solved numerically within the DFPT framework. In the DFPT approach, the many-body problem is simplified by a single-particle approximation; then, the single-particle wave functions are expressed as a linear combination of predefined basis functions. Thus, we can obtain a matrix equation to be solved numerically. The basis functions can be plane-waves, as used in Quantum ESPRESSO [4], VASP [5], and ABINIT [6], or mixed Gaussian and plane-wave as used in CP2K [7]. Although the above plane-wave basis sets can be converged systematically, the oscillatory behavior near the atomic nucleus cannot be accurately represented because of the excessive computational demands (e.g., 10^5 plane waves are needed for one core orbital). As a result, when using these basis sets, pseudization methods [8] using pseudo-potentials or projector-augmented waves have been introduced, in which the core potential is replaced by a "fake" one. Although the pseudo-potentials have been carefully constructed to keep the valence part consistent with the all-electron method, core-shell information remains missing. In order to consider the core and valence states on an equal footing and achieve better precision compared with the pseudization method [8], all-electron approaches have been developed, e.g., the all-electron Gaussian atomic orbital method in Gaussian [9], CRYSTAL [10], and the all-electron numerical atomic orbitals method in Dmol [11] and FHI-aims [1].

So far, most DFPT studies of physical properties (e.g., Raman spectra) have focused on systems with hundreds of atoms because

of the high computational demands involved. To the best of our knowledge, the largest full *ab initio* Raman spectra simulation was around 400 atoms [12, 13] using a Gaussian basis set. Therefore, a much larger system is needed to access new physical phenomena. The fragment method [14–16] was proposed in order to deal with large systems (around 1000 atoms); however, the fragments need to be carefully chosen to reduce interpretation errors [17, 18] and extended to the whole system for the simulation of very delocalized modes in the supercell [16].

Overall, the current state-of-the-art DFPT methods for large-scale systems are limited with respect to either accuracy or computational efficiency and scalability. Thus, DFPT calculations on large-scale systems have been severely limited. This represents the key bottleneck in simulating the physical properties of real systems.

In this work, we use the DFPT modules implemented in the all-electron full-potential massively scalable FHI-aims package to demonstrate that we can perform full QM calculations of Raman spectra of real biological systems containing up to 3006 atoms through massively parallel new algorithms on the new-generation Sunway supercomputer, without introducing approximations to the QM beyond those inherent to DFT/DFPT itself.

2.2 HPC System and Environment

The new-generation Sunway supercomputer is used for performance assessment in this work, which is the successor of the Sunway TaihuLight supercomputer. Similar to the Sunway TaihuLight system, the new Sunway supercomputer adopts a new-generation of domestic high-performance heterogeneous many-core processors and interconnection network chips in China.

The new SW processor is designed for massive thread and data parallelism and to deliver high performance on parallel workloads. The architecture of the SW26010Pro processor is shown in Fig. 1. Each processor contains 6 core-groups (CGs), with 65 cores in each CG, and in total 390 cores. Each CG contains one management processing element (MPE), one cluster of computing processing elements (CPEs) and one memory controller (MC). The MPE within each CG is used for computations, management and communication. The CPEs is organized as an 8×8 mesh (64 cores) and is designed to maximize the aggregated computing power and to minimize the complexity of the micro-architecture. The CPEs are organized with a mesh network to achieve high-bandwidth data communication (P2P and collective communications) among the CPEs in one CG, which is called remote scratchpad memory access (RMA).

Each SW26010Pro processor contains 96 GB memory, with 16 GB memory in each CG. The MPE and the CPEs within the same CG share the same memory which is controlled by the MC. Each CPE has a 32 KB L1 instruction cache, and a 256 KB scratch pad memory (SPM, also called the Local Data Memory (LDM)), which serves the same function as the L1 cache. The data storage space can also be configured as a local data cache (LDCache) which is automatically managed by the hardware. Data transfer between LDM and main memory can be realized by direct memory access (DMA), and data transfer between LDCache and main memory can also be realized by conventional load/storage instructions. The CPE adopts the SW64 instructions and provide 512-bit SIMD support. All

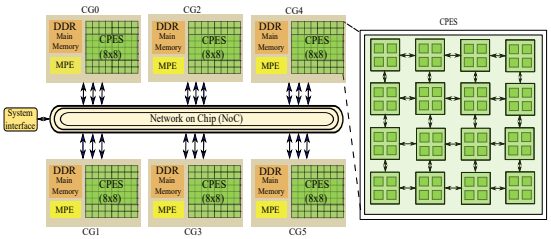

Figure 1: The architecture of the SW26010Pro processor.

the double precision, single precision, semi-precision floating-point computation and integer computation are supported.

2.3 The *ab initio* Raman spectra simulation

In the subsequent section, we present our simulation method for *ab initio* Raman spectra using the DFPT method in FHI-aims.

Before the DFPT calculation, the normal DFT calculation needs to be performed to solve the Kohn–Sham (KS) single-particle equations:

$$\hat{h}_{\mathrm{KS}}\psi_i = \left[\hat{t}_s + \hat{v}_{ext}(r) + \hat{v}_H + \hat{v}_{xc}\right]\psi_i = \epsilon_i\psi_i \,, \tag{1}$$

where the first term on the left-hand side is the KS Hamiltonian ($\hat{h}_{KS}$), $\hat{t}_s$ represents the single particle kinetic operator of the electron, $\hat{v}_{ext}$ is the (external) electron-nuclear potential, $\hat{v}_H$ is the Hartree potential, and $\hat{v}_{xc}$ represents the exchange-correlation potential. The KS single-particle states ψ_i and their eigenenergies ϵ_i can be calculated by solving Eq. (1). If an external electric field $\mathbf{E} = (e_x, e_y, e_z)$ with strengths e_γ is applied to the system, the KS Hamiltonian gains an additional term, $\hat{h}^{(1)}_{KS} = \hat{h}_E = -\mathbf{r}\cdot\mathbf{E}$. Thus, we obtain a perturbative version of the KS equation, which is called the Sternheimer equation:

$$(\hat{h}_{KS} - \epsilon_i)\,|\psi_i^{(1)}\rangle = -(\hat{h}^{(1)}_{KS} - \epsilon_i^{(1)})\,|\psi_i\rangle \ . \tag{2}$$

This equation can be solved self-consistently, yielding the corresponding first-order density, where f_i denotes the occupation number of eigenstate ψ_i:

$$n(\mathbf{r})^{(1)} = \frac{\partial n^{(0)}(\mathbf{r})}{\partial e_\delta} = \sum_i f_i \left[\psi_i^{*(0)}(\mathbf{r})\psi_i^{(1)}(\mathbf{r}) + \psi_i^{*(1)}(\mathbf{r})\psi_i^{(0)}(\mathbf{r})\right]. \tag{3}$$

Then, we obtain the polarizability, which corresponds to the second-order derivative of the total energy with respect to the external electric field:

$$\alpha_{\gamma\delta} = \int r_\gamma . \frac{\partial n^{(0)}(\vec{r})}{\partial e_\delta} d\vec{r}. \tag{4}$$

After the polarizabilities (α) have been determined using the DFPT approach, the Raman intensity is proportional to the square of the third-order derivative of the total energy with respect to both the external electric field and the atomic displacement:

$$I_{\mathrm{Raman}}(\omega_p) \propto (\alpha'_{ij})^2_p, \tag{5}$$

where $(\alpha'_{ij})_p = (\partial\alpha_{ij}/\partial Q_p) = \left(\sum_I e_{I,p}\partial\alpha_{ij}/\partial R_I\right)$ is the derivative of the ij component of the polarizability with respect to the

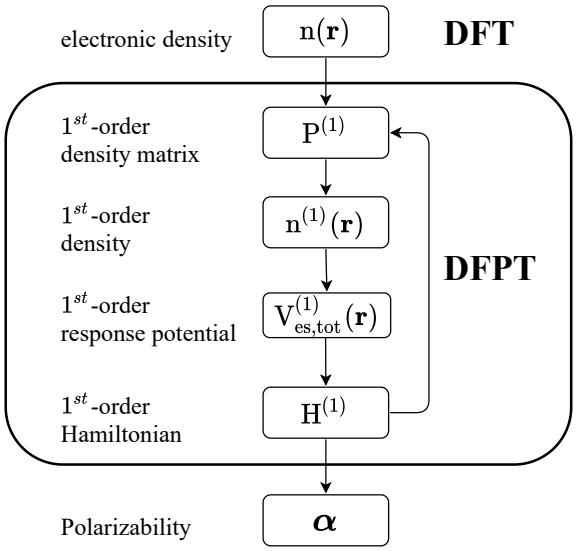

Figure 2: Flowchart for density-functional perturbation theory (DFPT) with the all-electron approach.

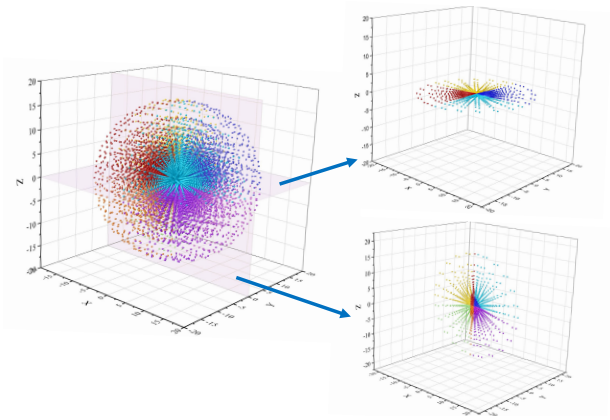

Figure 3: Three-dimensional illustration of the grid distribution of the hydrogen molecule (H_2). The integration points are distributed into batches, which are labeled with different colors.

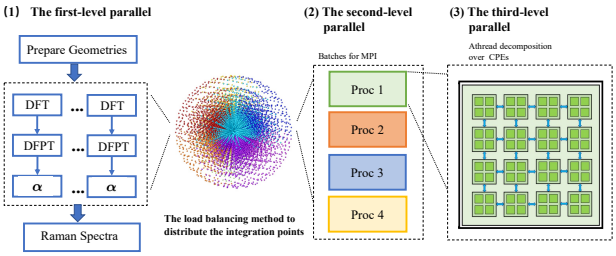

Figure 4: Three-level parallelization strategy for Raman spectra simulation with density-functional perturbation theory (DFPT). (1) The first-level parallel is over different geometries. (2) Process-level parallelization for grid integration, distributed over processed and labeled with different colors. (3) Thread-level parallelization for grid integration; the batches are then distributed on the threads to provide many-core acceleration.

displacement of normal phonon mode e_p. These derivatives are computed using finite differences; we evaluate the polarizability tensor from DFPT [3] at $6N$ ($3N$ forward and $3N$ backward) nuclear displacements in the unit cell around the equilibrium position, where N is the number of atoms per unit cell. Then, the above atomic derivatives are multiplied by the phonon eigenvectors to give $(\alpha'_{ij})_p$ for the calculation of Raman intensity.

A flowchart of the DFPT in FHI-aims is shown in Fig. 2. After the ground state calculation with DFT has been completed, the response of the overlap matrix is calculated. Then, the DFPT cycle begins, using an initial guess for the response of the density matrix $P^{(1)}$, which then allows the respective density $n^{(1)}(\mathbf{r})$ to be constructed. The associated response of the electrostatic potential $V^{(1)}_{es,tot}(\mathbf{r})$ is calculated by solving the Poisson equation in real space. The response Hamiltonian $H^{(1)}$ is calculated with the response density and potential. In turn, all these ingredients enable us to set up the Sternheimer equation to get the response density matrix $P^{(1)}$. We iteratively repeat the DFPT loop until self-consistency is reached, i.e., until the changes in $P^{(1)}$ become smaller than a user-defined threshold. Finally, the physical properties are evaluated using the converged response density matrix. It should be noted that, we speedup the convergence of the DFPT calculation with the direct inversion of the iterative subspace (DIIS) method [19], which is used to find a good approximation of the final solution as a linear combination of a set of trial vectors generated during an iterative solution of a problem.

3 INNOVATIONS

3.1 All-electron grids and the 3 level parallelization strategy

FHI-aims uses all-electron all-potential numerical atomic basis functions to achieve high-precision results. In this scheme, the all-electron atomic orbitals are discretized using an atom-centered grid [20], as illustrated in Fig. 3, in order to treat all-electron full-potential systems where the integrand is dominated by cusps at atomic nuclei. This atom-centered grid is first partitioned for each atom, and then the single-center (atom) grids are further separated into radial and angular parts, such that radially the atom-centered grid consists of several spherical integration shells with radial integration weights w_{rad}[1, 21]. On these shells, angular integration points are distributed such that spherical harmonics up to a certain order can be integrated exactly using the Lebedev grids[22], with angular integration weights w_{ang}. In this work, all the perturbation

properties are calculated within such discretized three-dimensional physical grids, which are suitable for massively parallel implementations.

In order to partition non-even grid meshes in an efficient way, the grid-adapted cut-plane method is used to form batches [23], as indicated by the different colors in Fig. 3. The procedure to obtain batches is as follows: first, the centers of mass for all points are computed; second, the direction of the cut-plane is determined by computing the normal of the plane; third, the position of the cut-plane is computed to divide all the points into two even-sized sets, and the full points are split into two subsets (batches) using the cut-plane. The above procedure is repeated until all the batches are the desired size (around 100 to 300 points per batch).

The load balancing for integration is achieved by eventually distributing the integration points over the MPI processes, as shown in Algorithm 1. The batches/grids are distributed according to the current summation of the points in each process; the new batch is always sent to the process with the minimal number of points. In this way, load balancing for grid integration is achieved. During an integration calculation, e.g., that of the response Hamiltonian matrix, each batch is calculated locally for one part of the matrix element, and finally the matrix elements from all processes are combined.

Algorithm 1 Load-balancing method to distribute integration points.

In:
n_{batches}: total number of batches in the integration
n_{proc}: total number of MPI processes

1: **for** $i \leftarrow 1, n_{\text{batches}}$ **do**
2: **for** $j \leftarrow 1, n_{\text{proc}}$ **do**
3: update the total number of points N^{j}_{points} in the j^{th} process
4: **end for**
5: find the minimal $N^{j_{\min}}_{\text{points}}$ and label the $j_{\min}$ process
6: add all points in the batch i to the $j_{\min}$ process
7: **end for**

Three levels of parallelization are used for Raman spectra calculations. As shown in Fig. 4, the calculation of the polarizability can be performed in an embarrassingly parallel manner with different geometries, that is, the first level of parallelization. As no communication between the polarizability calculations is required, we split the whole CPU pool into different sub-groups and sub-communicators. Within each sub-group, the polarizability is calculated with the DFPT method. Within each DFPT calculation, another two levels of parallelization have been adopted for calculation of the numerical integration. The first level of parallelization for DFPT is performed over the batches, which are distributed across all MPI processes. This enables good parallel scalability using the adapted batch distribution algorithm to achieve load balance. The second level of parallelization for DFPT is again performed over the batches within one process. Acceleration on threads can further improve the performance.

3.2 Optimization of the response density and response potential

The response density can be written as the following matrix multiplication:

$$n^{(1)}(\mathbf{r}) = \sum_{\mu,\nu} P^{(1)}_{\mu,\nu} \chi^{(0)}_{\mu}(\mathbf{r}) \chi^{(0)}_{\nu}(\mathbf{r}), \tag{6}$$

where $P^{(1)}_{\mu m,\nu n}$ is the response density matrix and $\chi^{(0)}_{\nu}$ is the wave function; μ,ν refer to the atomic orbit index.

The corresponding response potential ($V^{(1)}(\mathbf{r})$) can be evaluated from the electronic response density ($n^{(1)}(\mathbf{r'})$) by the following integration:

$$V^{(1)}(\mathbf{r}) = \int d\mathbf{r'} \frac{n^{(1)}(\mathbf{r'})}{|\mathbf{r} - \mathbf{r'}|}, \tag{7}$$

which is equivalent to the solution of the Poisson equation $\nabla^2 V^{(1)}(\mathbf{r}) = -4\pi n^{(1)}(\mathbf{r})$.

The response potential can be evaluated with the multipole Ewald [24] method and can be written as

$$V^{(1)}(r) \quad = \quad V^{\text{MP}}(r) + V^{\text{Ewald}}_{\text{real}}(r) + V^{\text{Ewald}}_{\text{recip}}(r). \tag{8}$$

We have ported the computation kernels of both the response density and response potential into the new-generation Sunway processor. As the calculation of the response density and response potential are distributed over the grid points, there is no communication between the processes and threads during the computation. We then perform DMA tilings to efficiently access the main memory; we also adopt double buffering to overlap the computations with memory accesses. In the following, we use the calculation of the response potential to demonstrate our optimizations.

3.2.1 Static loop tiling. Loop tiling is an important transformation for exploiting the spatial and temporal locality of data accesses in loop nests. Previous work [25–28] has demonstrated the loop tiling model for such scratch-pad memory (SPM)-based architectures. The calculation of the response potential contains two kernels, with *kernel*1 calculating potential in real-space ($V^{\text{MP}}(r) + V^{\text{Ewald}}_{\text{real}}(r)$) and *kernel*2 ($V^{\text{Ewald}}_{\text{recip}}(r)$) updating potential in reciprocal-space. Therefore, we apply loop tiling for the two kernels individually.

For *kernel*1, we allocate 128KB SPM and apply static loop tiling to keep the blocks of regularly access the arrays, e.g. *coord*, as shown in Figure 5.

*kernel*2 is more complicated, as it involves both regular accesses, e.g., *'electrostatic coef'* which refers to the electric-static potential related coefficients, and irregular accesses, e.g., the result of complex multiply of dimension x and y (WP_{xy}). Therefore, we allocate 60KB SPM for regularly accesses, by applying static loop tiling, and allocate the remaining SPM to the irregularly accessed WP_{xy} ($WP_{xy}[k\_points\_es[1][n] \times k\_points_{max} + k\_points\_es[0][n]]$.

Although the access of WP_{xy} is irregular from the kernel side, but we can leverage the cross-host-kernel analysis [29] approach to determine its subscript. In particular, we can derive that the two-dimensional *k_points_es* is used to traverse the reciprocal space thus the access of WP_{xy} is continuous. Therefore, after the cross-host-kernel analysis is performed, we can use static tiling for the

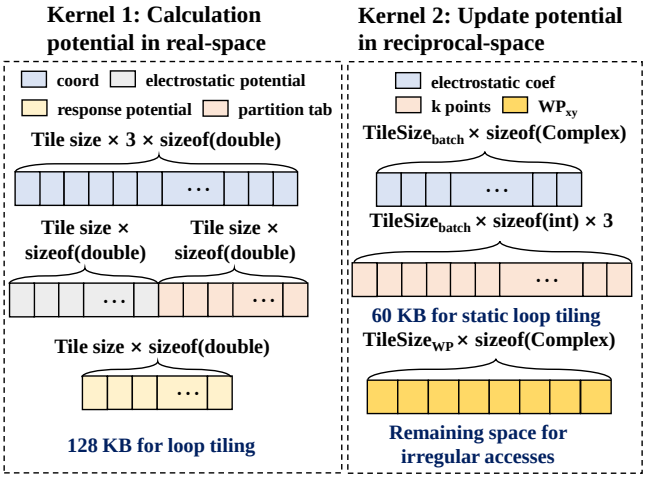

Figure 5: The illustration of loop tiling.

irregularly accessed array WP to fit each tile into the remaining SPM space.

3.2.2 Double buffering. Having determined the loop tiling strategy to utilize the on-chip SPM efficiently, we need to efficiently fetch data from the main memory to the SPM. A computing processing element (CPE) has two execution pipelines ($P0$ and $P1$). $P0$ supports scalar and vectorized computing operations of both floating-point and integer types, whereas $P1$ supports scalar and vectorized data load/store, compare and jump operations, and scalar integer operations. The double pipelines provide an opportunity for the overlapping of data access and computation operations [30]. For this purpose, we leverage double buffering [31] to overlap the data transfer and the on-chip computation.

The double buffering process is shown in Fig. 6. When double buffering is not used, the data fetching and computation are performed in a sequential way, as shown in the top panel of the figure. By comparison, we introduce two equal-sized SPM buffers, as shown in the bottom panel of Fig. 6; thus, the computation in one buffer and the DMA data fetching to another buffer can be executed simultaneously.

3.2.3 Vectorization. To further improve the computation efficiency of CPEs, response potential calculation exploits the vectorization opportunity to the functions in $kernel1$, e.g., cubic spline interpolation (CSI).

Algorithm 2 Compute multipole component By CSI

```
1: procedure CubicSpline(spline for electrostatic potential spl)
2:     prefetch s_{i/j/k/l} through DMA(spl)
3:     t1 ← i_r_log
4:     t2 ← i_r_log × i_r_log
5:     t3 ← i_r_log × i_r_log × i_r_log
6:     for all n ← 0, l_h_dim do
7:         multipole_component ← s_i + s_j × t1 + s_k × t2 + s_l × t3
8:     end for
9: end procedure
```

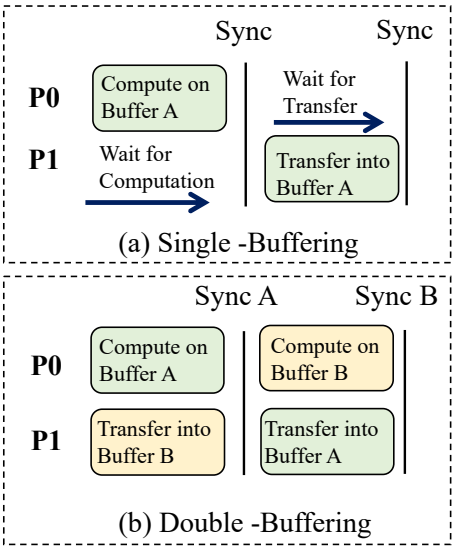

Figure 6: The illustration of double buffering optimization.

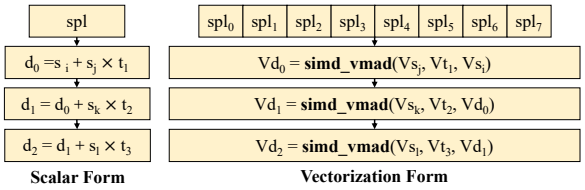

Figure 7: The illustration of vectorization for the CSI function.

The function of CSI is to calculate multipole_component, as shown in Algorithm 2. In particular, each spline interval pair maintains the its own spline parameters, and performs an element-wise computation to the spline array in the innermost loop nest. Therefore, the innermost loop nest can be vectorized, as shown in Figure 7, which compares the scalar and vector operations for the cubic spline interpolation. By employing the 512-bit SIMD operations, we can exploit data-level parallelism and completes the computation for 8 splines in one iteration.

3.3 Optimization of the response Hamiltonian

The calculation of the response Hamiltonian is an integration over the grid as follows:

$$H^{(1)}_{\mu,\nu} = \int \chi_\mu(\mathbf{r})\hat{h}_{KS}\chi_\nu(\mathbf{r})d\mathbf{r}. \tag{9}$$

After evaluating the integration over every CPE, the Hamiltonian elements are summed up using a large array reduction, which can be written as follows:

$$arr[idx] += val, \tag{10}$$

where arr is a large array, and val is irregularly distributed. In practical applications, the shape of the above reduction operation will often occur; because the capacity of local data memory (LDM) is limited, the size of array arr exceeds the LDM capacity, and the

traditional specification method is no longer applicable. However, if *arr* units in main memory are updated directly, write conflicts between CPEs will occur, and CPE locks need to be used to ensure that the write operation is correct. In this case, 64 CPEs contend for the same lock and *arr* to access main memory directly, which can seriously affect the performance of the reduction operation. Therefore, we designed an efficient distributed reduction based on the RMA communication mechanism between CPEs. The principle of this method is as follows: the large array *arr* that needs to be contracted is divided into 64 pieces of data, and 64 CPEs are responsible for the contracted operation for each piece of data, respectively. Each CPE applies a temporary buffer buf on LDM to buffer the large data blocks for which it is responsible. Besides, each slave core creates 64 send buffer units (denoted S0, S1, ... , S63) and 64 receive buffer units (denoted R0, R1, ... , R63), which are used to send and receive the RMA messages of 64 slave cores corresponding to the reduction operation, respectively. The reduction operation is performed as follows.

- Step 1: According to the current *idx* value, calculate the location of the element to be updated in the CPE managed data interval (assuming the destination CPE ID is n), and then cache the reduction value and the reduction operator to the local sending cache of the corresponding CPE's buffer unit S_n.
- Step 2: Check whether the sending buffer unit S_n is full. If so, send the contents of S_n to CPE n using RMA, and reset the sending buffer unit S_n.
- Step 3: Poll to process its own receiving buffer unit. If valid data is found in the receiving buffer unit, the reduction value, location, and other information of the receiving buffer unit will be resolved.
- Steps 4 and 5: If the original data of the reduction location are found to have been buffered into buf, the update operation will be performed directly; otherwise, the buffered buf data will be brushed back to main memory first, and then the required data fragments will be buffered into buf with DMA.

This algorithm cleverly utilizes the idea of cache references to map the big data that are waiting for reduction into blocks in LDM, and then uses an RMA communication mechanism between CPEs to aggregate and batch the reduction information, so that the communication bandwidth between CPEs can be efficiently utilized. This method has been integrated into the Sunway-OpenACC. The performance of the calculation of the Hamiltonian has been greatly improved by using this method to solve the discrete reduction problem of the response Hamiltonian for large arrays.

3.4 Optimization of the MPI Allreduce operation

For high-efficiency implementation of MPI Allreduce with large amounts of data, it is usually necessary to take into account both communication and calculation performance. The "Reduce-Scatter followed by Allgather" is a typical method, it reduces the total communication amount, while calculation is both overlapped within communication and parallelized among the MPEs during the process of "Reduce-Scatter" as the data are divided into blocks. However, this implementation involves two main challenges. On the one

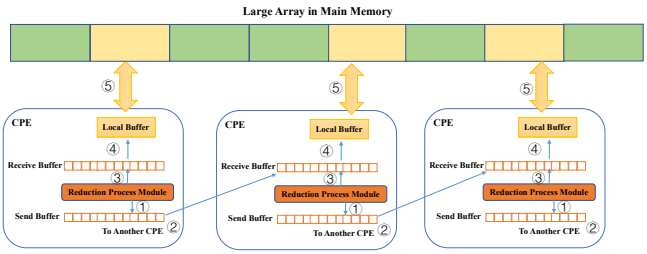

Figure 8: Schematic diagram of large arrays reduction algorithm.

hand, as the length of data increases, the proportion of calculation continues to increase, and it becomes difficult to completely hide the calculation. On the other hand, synchronous mode in which the MPE is responsible for communication or task scheduling while the SPE is responsible for calculation, may causes idleness of the CPE , that the powerful computing capability cannot be fully utilized.

In order to overcome these problems, we propose a MPI Allreduce optimization method integrating heterogeneous architecture contrary to large amounts of calculation. This method combines aggregation of memory access and concurrent computing of the CPEs to accelerate Allreduce, avoid idle resources in synchronous execution mode, and implement deep integration of Allreduce and heterogeneous architecture. When the amount of data is relatively small, the idle CPEs directly access main memory for calculation, effectively utilizing the concurrent computing capabilities. When the amount of data is large, the local memory of the CPE (the private memory of each CPE with low capacity but fast speed, referred to as LDM) is introduced, into which the data located in the main memory are read in batches, followed by concurrent computing; the results are then written back to main memory. Different methods are adopted, mainly owing to the startup overhead of batch transmission. Direct use of fine-grained memory access under small data volume conditions can avoid the startup overhead of batch transmission. Using batch transmission in the case of a large data volume takes advantage of the bandwidth of main memory and improves the execution efficiency of large amounts of access from the CPEs. Algorithm 3 shows pipelined concurrency of batch transmission and calculation. This method is based on the asynchronous batch transmission function of the CPE (lines 10, 11, 21, 22, 24, 36), which uses a "reply word"; every time data transmission is completed, the value of the reply word will be increased by 1. Moreover, a double buffer mode is needed to achieve pipelined concurrency. Based on the value of the reply word (lines 17, 30), the reading and writing of the two (A/B) buffers can be accurately scheduled. As the Allreduece operation needs to read both the source space and the destination space, calculate them, and then write the results back into the destination space, the occupied buffer of LDM space needs to be divided into four parts (line 3), of which blocks 0 and 1 are used as buffer A (line 6), and blocks 2 and 3 are used as buffer B (line 7). Through asynchronous batch transmission, the read of buffer B (lines 21, 22) and the calculation of buffer A (line 23) are overlapped, thereby hiding the overhead of the calculation of buffer A.

The above optimization can efficiently use idle computing capability in synchronous concurrency mode, so as to accelerate the calculation of Allreduce collectives. Moreover, pipelined concurrency of batch transmission and calculation can fully hide the overhead of calculation and can effectively improve the performance of reduction collectives.

Algorithm 3 Pipelined Calculation on CPEs for MPI Allreduce.

In: $Src, Dst, Dtype, Cnt, Op, Ldm\_buf, Ldm\_buf\_sz$
Out: Dst

1: **function** REDUCE_LOCAL
2: $(Src, Dst, Dtype, Cnt, Op, Ldm\_buf, Ldm\_buf\_sz)$
3: $blk\_sz \leftarrow Ldmbuf\_sz/4/sizeof(Dtype) * sizeof(Dtype)$
4: $blks \leftarrow Cnt * sizeof(Dtype)/blk\_sz$
5: $rply \leftarrow 1$
6: $cur \leftarrow Ldm\_buf$
7: $next \leftarrow Ldm\_buf + 2 * blk\_sz$
8:
9: **if** $blks$ **then**
10: CPE_MEMCPY_IN_ASYNC$(cur, Dst, blk_sz, \&rply)$
11: CPE_MEMCPY_IN_ASYNC$(cur + blk_sz, Src, blk_sz, \&rply)$
12: $transferred \leftarrow transferred + blk\_sz$
13: $i \leftarrow i + 1$
14: **end if**
15:
16: **while** $transferred < blks * blk\_sz$ **do**
17: **while** $rply < 3 * i$ **do**
18: **end while**
19: $tmpdst \leftarrow Dst + transferred$
20: $tmpsrc \leftarrow Src + transferred$
21: CPE_MEMCPY_IN_ASYNC$(next, tmpdst, blk_sz, \&rply)$
22: CPE_MEMCPY_IN_ASYNC$(next + blk_sz, tmpsrc, blk_sz, \&rply)$
23: OP$(cur, cur + blk_sz, blk_sz/sizeof(Dtype))$
24: CPE_MEMCPY_OUT_ASYNC$(tmpdst - blk_sz, cur, blk_sz, \&rply)$
25: $transferred \leftarrow transferred + blk\_sz$
26: $i \leftarrow i + 1$
27: $next \leftrightarrow cur$
28: **end while**
29:
30: **while** $rply < 3 * i$ **do**
31: **end while**
32:
33: **if** $blks$ **then**
34: OP$(cur, cur + blk_sz, blk_sz/sizeof(Dtype))$
35: $tmpdst \leftarrow Dst + transferred$
36: CPE_MEMCPY_OUT$(tmpdst - blk_sz, cur, blk_sz)$
37: **end if**
38:
39: **end function**

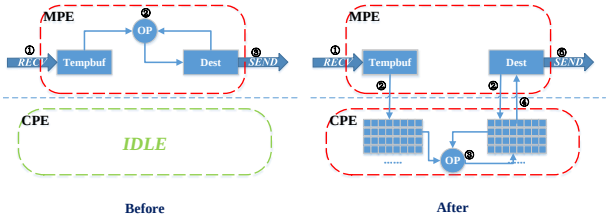

Figure 9: The optimization for the MPI Allreduce operation on the heterogeneous many-core architecture.

3.5 Portability of Our Optimizations

The multi-level parallelization scheme is general so it can be applied to other platforms. The loop tiling is exploited to optimize bandwidth, in order to extend the tiling strategies for other many-core architectures not supported DMA, dynamic tiling might be triggered to adjust tile sizes at runtime for irregular memory accesses to boost performance. Our vectorization is also portable to any platform with SIMD unit. We only need to re-design the mask code splicing by the platform's SIMD shuffle instruction and vector width. The double buffering method can be applicable and beneficial on other manycore architectures.

4 EVALUATION

4.1 Simulation Validation

The dielectric constant is a measure of the amount of electric potential energy, in the form of induced polarization that is stored in a given volume of material under the action of an electric field, it is directly related with the polarizability, which is defined as

$$\varepsilon^{\infty}_{\gamma\delta} = \delta_{\gamma\delta} + \frac{4\pi}{V_{uc}}\alpha_{\gamma\delta}. \tag{11}$$

in which, V_{uc} refers to the volume of the unit cell.

In order to validate our implementation, we have calculated the dielectric constant of 19 zinc blende semiconductors. Here a comparison of our results with plane wave code QUANTUM ESPRESSO. In all cases, the calculations were performed for the equilibrium geometry determined by relaxation (maximum force $< 10^{-4}$ eV/Å), at LDA level of theory using fully converged numerical parameters, i.e., by using 16×16×16 **k**-points in the primitive unit cell, "tier 2" basis sets and so called "really tight" defaults for the integration grids in FHI-AIMS; And by using an energy cut-off of 50 Hartree for the truncation of the plane wave basis set, and Troullier-Martins norm-conserving pseudopotentials (without nonlinear core corrections) in Quantum Espresso. Fig. 10 illustrates the excellent agreement between our results and the one given by Quantum Espresso code, the mean relative error for the dielectric constants between FHI-aims and Quantum Espresso is within 1% for the 19 examples. It should be noted that, the above good agreement comes from only considering **s** and **p** electrons as valence shells that the pseudopotential can work well. When we consider the system containing **d** electrons and include it in the calculation, such as GaSb, the all-electron scheme can improve the calculation accuracy by 15% compared with the pseudopotential method [3].

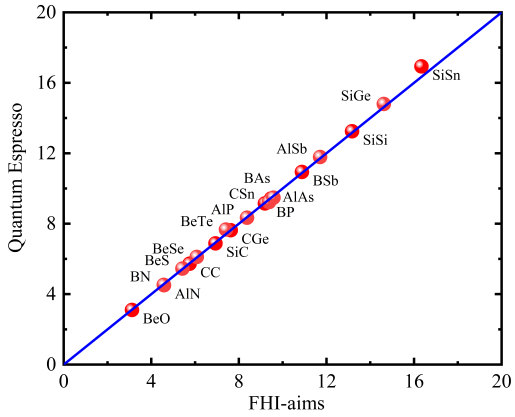

Figure 10: The comparison of the high-frequency dielectric constants of various semiconductors between our all-electron approach and the results calculated with Quantum Espresso.

To further validate our simulations, we turn to a comparison of our results for the Raman spectrum of the H_2O molecule with those obtained using the Gaussian code. Fig. 11 shows our computed harmonic Raman frequency and intensity of H_2O (tight, tier3), compared with the values calculated with Gaussian (aug-cc-pVDZ). Both spectra are calculated with local density approximation (LDA) functions. Our results for the Raman spectrum of the H_2O molecule in the O–H stretching region show very good agreement with those from Gaussian, and the relative errors are within 0.5%.

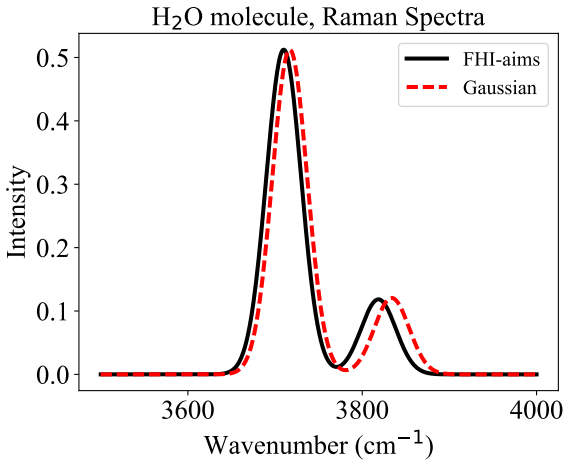

Figure 11: Comparison of the frequencies and intensity of Raman spectra between FHI-aims and Gaussian. The relative errors between FHI-aims (tier2) and Gaussian (aug-cc-pVDZ) are within 0.5% in this example. The units of frequency and intensity are cm^{-1} and Å^4/amu, respectively

Table 1: Case configurations for evaluation. Here the silicon solid is use.

case name		grid	basis	average points per batch
Si solid	#1	35836	18	100
	#2	56860	18	100
	#3	35836	36	100
	#4	56860	50	100
	#5	35836	36	200
	#6	35836	36	300

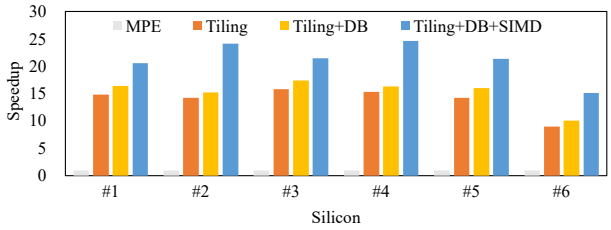

Figure 12: The performance comparison among different optimization steps for the response potential (V1) calculation.

4.2 Performance Speedup Results

First we use the calculation of response potential as the example to illustrate the performance improvements for different kinds of optimization methods on one new Sunway node which contains 390 cores (6 MPEs with 384 CPEs). Fig. 12 gives a comparison of different versions optimized by a series of methods. We use different settings (as shown in Tab. 1) to calculate solid silicon that the number of grid and the number of basis are changed. First, the DMA loop tiling method could gain the performance improvements, ranging from 10x to 15x by increasing the data locality. Second, double buffering applied after loop tiling is introduced to overlap the data fetching to SPM and the computation, the performance results shows that when double buffering is enabled, all the data fetching operations can be completely over-lapped with the computation, so we make the performance improvements to 16x. Third, we applied vectorization to further improve the instruction level paralelism, finally we achieve a performance speedup around 20x compared to the original MPE version.

Fig. 13 gives the speedups of different DFPT kernels within one Sunway processor with 6 CG. The performance speedups refer to the calculation with one MPE. We also use 6 examples of silicon solid with different parameters (e.g. number of basis sets and grid points as shown in Tab. 1) to perform the DFPT calculation. As shown in Fig. 13, the acceleration of response potential ($V^{(1)}$) does not depend on the number of basis function, since it only contains the grid points calculation, so calculation with denser grid points(#2 and #4) give about 7% higher speedup than the others. On the other hand, the speedups of response density ($n^{(1)}$)/response Hamiltonian ($H^{(1)}$) both depend on the basis set and grid point. Taking $n^{(1)}$ as an example, the highest speedup ratio appears at #4 with the largest number of grid points (56860) and number of basis set (50). It is also found that the speedup also relate to the average number of

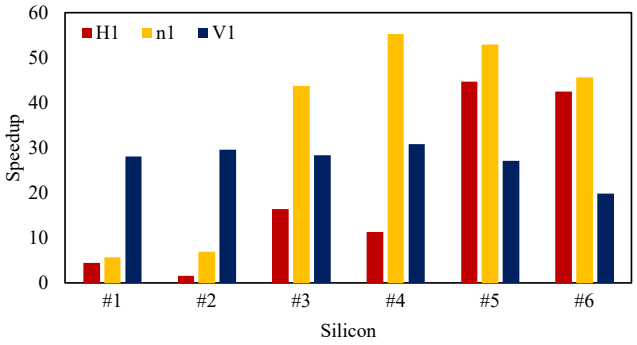

Figure 13: The performance speedups of different kernels within the DFPT calculation. The blue/yellow/red bars refer to the speedup of the response potential (V1)/response density (n1)/response Hamiltonian (H1) respectively.

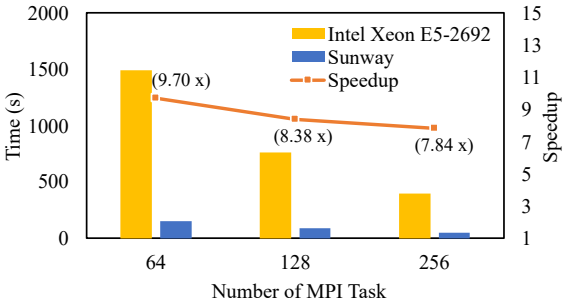

Figure 14: The performance comparison of the total time per DFPT iteration for RBD protein contained 3006 atoms.

the points per batch, by comparing the results of #3 #5 #6, we can see 200 points per batch show the highest acceleration.

In Fig. 14, we further evaluate the performance improvements over the CPU calculations which are performed on Tianhe-2 with Intel CPU (Xeon E5 2692 V2). We use the receptor binding domain (RBD) protein containing 3006 atoms as the example. All calculations use light settings and the LDA functional in FHI-AIMS. Thanks to the innovations in Sec. 3, the DFPT total time per cycle exhibits an overall 9.7x to 7.8x speedup with the number of process increasing from 64 to 256 compared with the CPU implementation.

Fig. 15 shows the speedup of the MPI Allreduce optimization for response potential, the speedup ratio reaches 2.2 to 2.6 when increasing the MPI processes from 256 to 1024. The increasing of the speedup radio can be understand as following: Suppose the data amount is L, and the number of processes is N, then $\log_2 N$ times of communication with length of $L/2^i$ are required during the implementation of Allreduce. Since the data after each communication step needs to be calculated, and computation volume is $\sum_{i=1}^{log_2 N} L/2^i = (1 - \frac{1}{N})L$, which is increasing when the number of processes N increases, so the speedup radio will also increase.

In Fig. 16, we compared the DFPT calculation performed with FHI-aims and Gaussian code. In FHI-aims, we use light setting for grid integration and tier 1 basis set, for Gaussian, we use Int=FineGrid

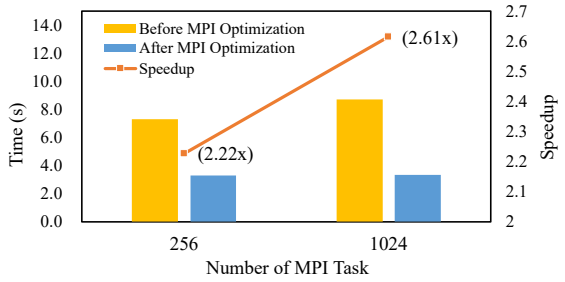

Figure 15: The optimization of the MPI Allreduce time during the response potential calculation of RBD protein contained 3006 atoms.

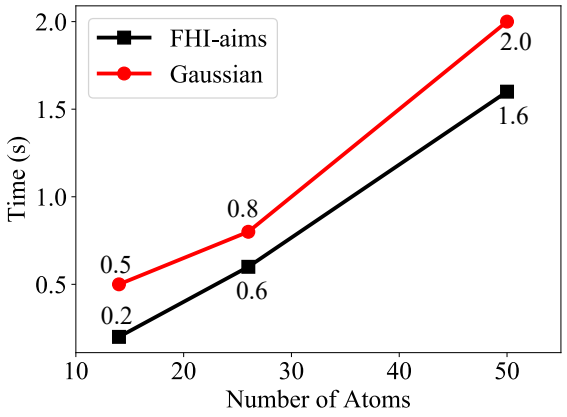

Figure 16: The performance comparison between FHI-aims and Gaussian for the time per DFPT iteration with three isolated H(C2H4)$_n$H molecules. All calculation are performed at Tianhe-2 with 12 processes.

and 6-31G** basis set to keep consistent. All calculation are performed at Tianhe-2 with 12 MPI tasks. We can see that for the DFPT calculation, FHI-aims is around 2.27x to 1.25x faster compared with Gaussian when the number of atoms in H(C2H4)$_n$H molecules changing from 14 to 50. For large systems such as RBD, Gaussian will run out of memory.

4.3 Scalability Results

In Fig. 17, we measure the runtime and strong scaling for the Raman spectra calculation of RBD protein complex. In all calculations, each MPI sub-group was mapped to 256 processes, and the total number of the polarizabilities are set to be 1175 for the strong scaling calculation. The code shows a good strong scaling performance, that with the number of the Sunway processes increases from 10,240 all the way to 300,800 for the calculation (the number of cores changing from 665,600 to 19,552,000), the parallel efficiency is > 80% with 300,800 processes, with a 25× speedup with respect to the 10,240 processes run.

Figure 18 shows the weak scaling for the Raman spectra calculation of RBD protein. In order to obtain the weak scaling data,

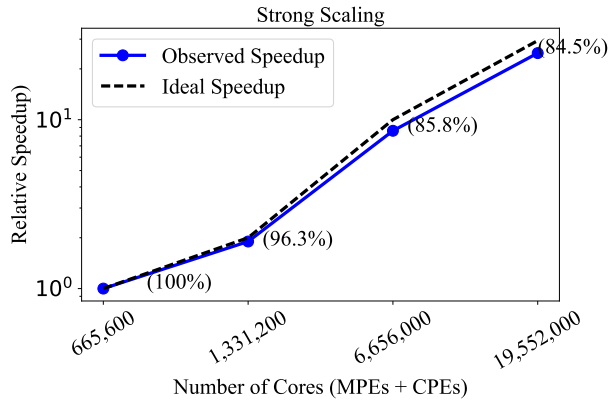

Figure 17: The strong scalability for the Raman spectra computation time of the RBD complex contained 3006 atoms. The blue line is the relative speedup and the black dash line is the ideal speedup. Parallel efficiency values are annotated in the parentheses.

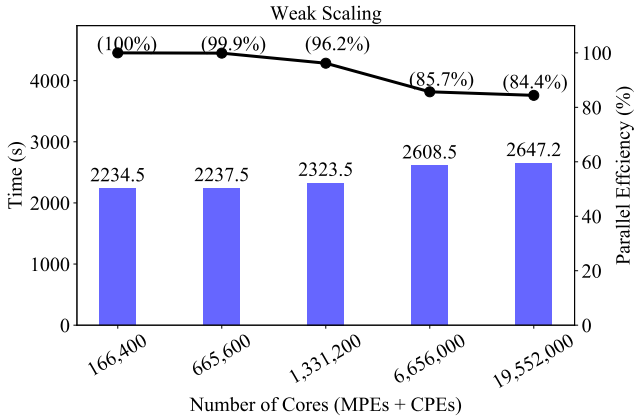

Figure 18: The weak scalability for the Raman spectra computation time of the RBD complex contained 3006 atoms. Here the number of the polarizabilities in the Raman spectra calculation are increased with the number of the cores (MPEs + CPEs). The blue bars is the simulation time, the black line is the weak parallel efficiency. Simulation time and parallel efficiency values are annotated on the top of bars and in the parentheses respectively.

we timed both the DFT and DFPT calculation on the RBD protein complex with an increasing number of the polarizabilities. The Raman spectra calculation code achieves excellent weak scaling which keeps a high parallel efficiency with the growth of cores. With 2,560 processes, as a reference, we achieves a parallel efficiency of 84.4% with 300,800 processes (19,552,000 cores).

5 APPLICATIONS

Finally, we demonstrate the DFPT application in the large realistic biological system. In biological systems, the usage of the Raman spectroscopy is increasing rapidly [32, 33], because it can provide the chemical and composition information for proteins in essentially all physical states, it can also probe the structural changes in proteins resulted from the protein-ligand interactions.

The computations are based on the crystal structure of the RBD protein from Qihui Wang et al. [34] at a resolution of 2.5 Å, which protein data bank (PDB) file number is 6LZG. Hydrogen atoms of the protein (see PDB file 6LZG) were added by using the ambertools.

The ACE2 has been reported to be the cellular receptor for the SARS-CoV-2 coronavirus, who uses the RBD of the surface Spike glycoprotein (S protein) to engage ACE2. One of the promising therapeutic strategy is to design high-affinity inhibitors to SARS-CoV-2 Spike RBD to compete with ACE2 binding. The Raman spectrum of the RBD can thus give the structure information of the protein, which will allow us to get fundamental understanding about the interaction between atoms.

The Raman spectra for the RBD protein is calculated with LDA functional, "light" basis set. The whole Raman spectra and the comparison with experimental data is shown in Fig. 19. The smearing of the theoretical Raman spectra is set to 5 cm^{-1} expect the spectra within Amide III zone, which is not smeared in order to illustrate finer vibrational information. Overall, we can see a good agreement between the calculated and experimental Raman spectra, although the intensities have relatively small differ, characteristic patterns are easily visible. The S-S bond stretching bands are located in the 500-550 cm^{-1} region, which shows that there are S-S bridges in the protein, this bands are found in both experimental and calculated spectrum. The bands near 800 cm^{-1} observed in the experimental and theoretical spectrum are the aromatic aminoacids (Tyrosine) bands, which come from the in-plane breathing mode of the phenol ring. The Trp/Phe aromatic Raman signal around 1001 cm^{-1} and the Trp band at 1112 cm^{-1} (calculated at 1003 and 1117 cm^{-1}, respectively) are related to the breathing mode of the Phenylalanine, these distinct spectral features are reproduced well with simulation. The amide III spectral region (1200-1360 cm^{-1}) can be used to correlated to the amide I band (stretching vibration of C=O ,around 1650 cm^{-1}) in order to get some additional details to the amide I. The two bands in this amid III region observed in the experimental measurement with a higher Raman intensity than our calculation, so does the C=C stretching band. The remaining part of the experimental spectrum with bands at around 1650 cm-1 is also obtained in our calculation, and we get similar relative intensity of the band at 1650 cm-1 for the amide I band compared with in the measured spectrum.

6 CONCLUSION

The innovations realized in this work, make FHI-aims suitable to the future exascale machines. Three levels of parallelization have been adopted to utilize the nature of the physical problems and the many-core architecture. To the best of our knowledge, this is the first reported quantum perturbation calculation for Raman spectra that can scale to over nearly ten millions of cores. The multi-level parallelization scheme for Raman spectra simulation proposed

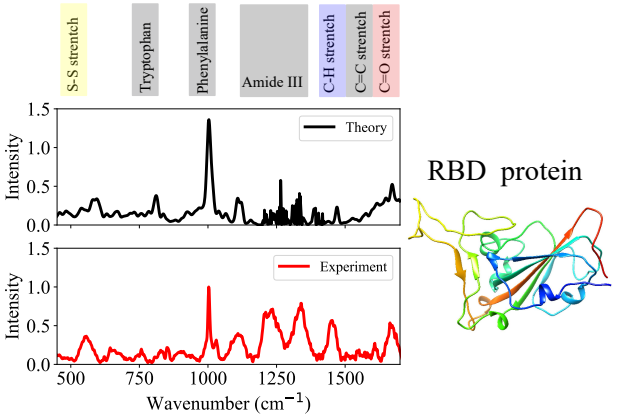

Figure 19: The simulated and experimental Raman spectra of the RBD protein, which demonstrate the sensitivity power of the Raman spectra in probing the protein structure.

in this work can be straightforwardly extended to other DFPT codes. The MPI optimizations, double buffering method, vector instructions method can also be adopted for other DFPT codes with the similar integration parts (response density, response potential, response Hamiltonian). Furthermore, the Raman spectra of the realistic biological system has been demonstrated to open new perspectives for applying the DFPT in biological systems. This work has also demonstrated the ability of the quantum mechanics approaches to simulate the protein system (SARS-CoV-2-RBD), which is starting point for using the quantum mechanics method to probe the structural change in proteins resulted from the protein-ligand interactions and opens up new possibilities for improving the accuracy of virtual drug screening.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (grant no. 22003073), by the State Key Laboratory of Computer Architecture Foundation under Grant No. CARCH 4205, CARCH 4411, by the National Natural Science Foundation of China (grant no. 62090024, 61872043, 61802368).

REFERENCES

[1] Volker Blum, Ralf Gehrke, Felix Hanke, Paula Havu, Ville Havu, Xinguo Ren, Karsten Reuter, and Matthias Scheffler. Ab initio molecular simulations with numeric atom-centered orbitals. *Comput. Phys. Commun.*, 180(11):2175–2196, November 2009.

[2] Honghui Shang, Christian Carbogno, Patrick Rinke, and Matthias Scheffler. Lattice dynamics calculations based on density-functional perturbation theory in real space. *Computer Physics Communications*, 215:26–46, jun 2017.

[3] Honghui Shang, Nathaniel Raimbault, Patrick Rinke, Matthias Scheffler, Mariana Rossi, and Christian Carbogno. All-electron, real-space perturbation theory for homogeneous electric fields: theory, implementation, and application within DFT. *New Journal of Physics*, 20(7):073040, jul 2018.

[4] Paolo Giannozzi, Stefano Baroni, Nicola Bonini, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Guido L Chiarotti, Matteo Cococcioni, Ismaila Dabo, Andrea Dal Corso, Stefano de Gironcoli, Stefano Fabris, Guido Fratesi, Ralph Gebauer, Uwe Gerstmann, Christos Gougoussis, Anton Kokalj, Michele Lazzeri, Layla Martin-Samos, Nicola Marzari, Francesco Mauri, Riccardo Mazzarello, Stefano Paolini, Alfredo Pasquarello, Lorenzo Paulatto, Carlo Sbraccia, Sandro Scandolo, Gabriele Sclauzero, Ari P Seitsonen, Alexander Smogunov, Paolo Umari, and Renata M Wentzcovitch. Quantum espresso: a modular and open-source software project for quantum simulations of materials. *Journal of Physics: Condensed Matter*, 21(39):395502 (19pp), 2009.

[5] G. Kresse and J. Hafner. Ab initio molecular dynamics for liquid metals. *Phys. Rev. B*, 47:558–561, Jan 1993.

[6] Aldo H. Romero, Douglas C. Allan, Bernard Amadon, Gabriel Antonius, Thomas Applencourt, Lucas Baguet, Jordan Bieder, François Bottin, Johann Bouchet, Eric Bousquet, Fabien Bruneval, Guillaume Brunin, Damien Caliste, Michel Côté, Jules Denier, Cyrus Dreyer, Philippe Ghosez, Matteo Giantomassi, Yannick Gillet, Olivier Gingras, Donald R. Hamann, Geoffroy Hautier, François Jollet, Gérald Jomard, Alexandre Martin, Henrique P.C. Miranda, Francesco Naccarato, Guido Petretto, Nicholas A. Pike, Valentin Planes, Sergei Prokhorenko, Tonatiuh Rangel, Fabio Ricci, Gian Marco Rignanese, Miquel Royo, Massimiliano Stengel, Marc Torrent, Michiel J. Van Setten, Benoit Van Troeye, Matthieu J. Verstraete, Julia Wiktor, Josef W. Zwanziger, and Xavier Gonze. ABINIT: Overview and focus on selected capabilities. *Journal of Chemical Physics*, 152(12), 2020.

[7] Thomas D. Kühne, Marcella Iannuzzi, Mauro Del Ben, Vladimir V. Rybkin, Patrick Seewald, Frederick Stein, Teodoro Laino, Rustam Z. Khaliullin, Ole Schütt, Florian Schiffmann, Dorothea Golze, Jan Wilhelm, Sergey Chulkov, Mohammad Hossein Bani-Hashemian, Valéry Weber, Urban Borštnik, Mathieu Taillefumier, Alice Shoshana Jakobovits, Alfio Lazzaro, Hans Pabst, Tiziano Müller, Robert Schade, Manuel Guidon, Samuel Andermatt, Nico Holmberg, Gregory K. Schenter, Anna Hehn, Augustin Bussy, Fabian Belleflamme, Gloria Tabacchi, Andreas Glöß, Michael Lass, Iain Bethune, Christopher J. Mundy, Christian Plessl, Matt Watkins, Joost VandeVondele, Matthias Krack, and Jürg Hutter. CP2K: An electronic structure and molecular dynamics software package -Quickstep: Efficient and accurate electronic structure calculations. *Journal of Chemical Physics*, 152(19), 2020.

[8] Kurt Lejaeghere, Gustav Bihlmayer, T. Bjorkman, Peter Blaha, S. Blugel, Volker Blum, Damien Caliste, Ivano E Castelli, Stewart J Clark, Andrea Dal Corso, Stefano de Gironcoli, Thierry Deutsch, John Kay Dewhurst, Igor Di Marco, Claudia Draxl, M. Du ak, Olle Eriksson, José A Flores-Livas, Kevin F Garrity, Luigi Genovese, Paolo Giannozzi, Matteo Giantomassi, Stefan Goedecker, Xavier Gonze, O. Granas, E K U Gross, Andris Gulans, François Gygi, D R Hamann, Phil J Hasnip, N A W Holzwarth, D. Iu an, Dominik B Jochym, François Jollet, Daniel Jones, Georg Kresse, Klaus Koepernik, E. Kucukbenli, Yaroslav O Kvashnin, Inka L M Locht, Sven Lubeck, Martijn Marsman, Nicola Marzari, Ulrike Nitzsche, L. Nordstrom, Taisuke Ozaki, Lorenzo Paulatto, Chris J Pickard, Ward Poelmans, Matt I J Probert, Keith Refson, Manuel Richter, G.-M. Rignanese, Santanu Saha, Matthias Scheffler, Martin Schlipf, Karlheinz Schwarz, Sangeeta Sharma, Francesca Tavazza, P. Thunstrom, Alexandre Tkatchenko, Marc Torrent, David Vanderbilt, Michiel J van Setten, Veronique Van Speybroeck, John M Wills, Jonathan R Yates, G.-X. Zhang, and Stefaan Cottenier. Reproducibility in density functional theory calculations of solids. *Science*, 351(6280):aad3000–aad3000, mar 2016.

[9] Michael Frisch, Martin Head-Gordon, and John Pople. Direct analytic scf second derivatives and electric field properties. *Chem. Phys.*, 141(2-3):189 – 196, 1990.

[10] Roberto Dovesi, Fabien Pascale, Bartolomeo Civalleri, Klaus Doll, Nicholas M. Harrison, Ian Bush, Philippe D'Arco, Yves Noël, Michel Rérat, Philippe Carbonnière, Mauro Causà, Simone Salustro, Valentina Lacivita, Bernard Kirtman, Anna Maria Ferrari, Francesco Silvio Gentile, Jacopo Baima, Mauro Ferrero, Raffaella Demichelis, and Marco De La Pierre. The CRYSTAL code, 1976-2020 and beyond, a long story. *The Journal of chemical physics*, 152(20):204111, 2020.

[11] B Delley. An all-electron numerical method for solving the local density functional for polyatomic molecules. *J. Chem. Phys.*, 92(1):508, 1990.

[12] Sandra Luber and Markus Reiher. Theoretical raman optical activity study of the ß domain of rat metallothionein. *Journal of Physical Chemistry B*, 114(2):1057–1063, 2010.

[13] Simone Salustro, Anna Maria Ferrari, Roberto Orlando, and Roberto Dovesi. Comparison between cluster and supercell approaches: the case of defects in diamond. *Theoretical Chemistry Accounts*, 136(4):1–13, 2017.

[14] Petr Bouř, Jana Sopková, Lucie Bednárová, Petr Maloň, and Timothy A Keiderling. Transfer of molecular property tensors in cartesian coordinates: A new algorithm for simulation of vibrational spectra. *Journal of Computational Chemistry*, 18(5):646–659, apr 1997.

[15] Shigeki Yamamoto and Petr Bouř. *Calculation of Vibrational Spectra of Large Molecules from Their Fragments*, pages 181–197. Springer Singapore, Singapore, 2018.

[16] Fabien Pascale, Simone Salustro, Anna Maria, Ferrari Michel, Rérat Philippe, and D Arco Roberto. The Infrared spectrum of very large (periodic) systems : global versus fragment strategies — the case of three defects in diamond. *Theoretical Chemistry Accounts*, 137(12):1–7, 2018.

[17] Noah S Bieler, Moritz P Haag, Christoph R Jacob, and Markus Reiher. Analysis of the Cartesian Tensor Transfer Method for Calculating Vibrational Spectra of Polypeptides. *Journal of Chemical Theory and Computation*, 7(6):1867–1881, jun 2011.

[18] Shigeki Yamamoto, Xiaojun Li, Kenneth Ruud, and Petr Bouř. Transferability of Various Molecular Property Tensors in Vibrational Spectroscopy. *Journal of Chemical Theory and Computation*, 8(3):977–985, mar 2012.

[19] Péter Pulay. Convergence acceleration of iterative sequences. the case of scf iteration. *Chem. Phys. Lett.*, 73(2):393–398, 1980.

[20] A. D. Becke. A multicenter numerical integration scheme for polyatomic molecules. *J. Chem. Phys.*, 88(4):2547–2553, 1988.

[21] Jon Baker, Jan Andzelm, Andrew Scheiner, and Bernard Delley. The effect of grid quality and weight derivatives in density functional calculations. *J. Chem. Phys.*, 101(10):8894–8902, 1994.

[22] Bernard Delley. High order integration schemes on the unit sphere. *J. Comput. Chem.*, 17(9):1152–1155, 1996.

[23] V. Havu, V. Blum, P. Havu, and M. Scheffler. Efficient integration for all-electron electronic structure calculation using numeric basis functions. *J. Comput. Phys.*, 228(22):8367–8379, December 2009.

[24] Bernard Delley. Fast calculation of electrostatics in crystals and large molecules. *The Journal of Physical Chemistry*, 100(15):6107–6110, 1996.

[25] M. S. Lam and M. Wolf. A data locality optimizing algorithm. In *Proceedings of the ACM SIGPLAN 1991 Conference on Programming Language Design and Implementation*, PLDI 91, pages 30–44, New York, NY, USA, 1991. ACM.

[26] S. Coleman and K. S. McKinley. Tile size selection using cache organization and data layout. In *Proceedings of the ACM SIGPLAN 1995 Conference on Programming Language Design and Implementation*, PLDI 95, pages 279–290, New York, NY, USA, 1995. ACM.

[27] J. Liu, Y. Zhang, W. Ding, and M. T. Kandemir. On-chip cache hierarchy-aware tile scheduling form ulticore machines. In *Proceedings of the 9th Annual IEEE/ACM International Symposium on Code Generation and Optimization*, CGO 11, page 161–170, Chamonix, France, 2011. ACM.

[28] S. Mehta, R. Garg, N. Trivedi, and P. Yew. Leveraging prefetching to boost performance of tiled codes. In *Proceedings of the 2016 International Conference on Supercomputing*, ISC 16, New York, NY, USA, 2016. ACM.

[29] Mingchuan Wu, Ying Liu, Huimin Cui, Qingfu Wei, Quanfeng Li, Limin Li, Fang Lv, Jingling Xue, and Xiaobing Feng. Bandwidth-aware loop tiling for dma-supported scratchpad memory. In *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*, PACT '20, page 97–109, New York, NY, USA, 2020. Association for Computing Machinery.

[30] W. Zhao, H. Fu, J. Fang, W. Zheng, L. Gan, and G. Yang. Optimizing convolutional neural networks on the sunway taihulight supercomputer. *ACM Transactions on Architecture and Code Optimization (TACO)*, 15(1):1–26, 2018.

[31] B. Khailany and B. Khailany. Cudadma: Optimizing gpu memory bandwidth via warp specialization. In *High Performance Computing, Networking, Storage and Analysis*, 2011.

[32] A. Rygula, K. Majzner, K. M. Marzec, A. Kaczor, M. Pilarczyk, and M. Baranska. Raman spectroscopy of proteins: A review. *Journal of Raman Spectroscopy*, 44(8):1061–1076, 2013.

[33] Holly J. Butler, Lorna Ashton, Benjamin Bird, Gianfelice Cinque, Kelly Curtis, Jennifer Dorney, Karen Esmonde-White, Nigel J. Fullwood, Benjamin Gardner, Pierre L. Martin-Hirsch, Michael J. Walsh, Martin R. McAinsh, Nicholas Stone, and Francis L. Martin. Using Raman spectroscopy to characterize biological materials. *Nature Protocols*, 11(4):664–687, 2016.

[34] Manli Wang, Ruiyuan Cao, Leike Zhang, Xinglou Yang, Jia Liu, Mingyue Xu, Zhengli Shi, Zhihong Hu, Wu Zhong, and Gengfu Xiao. Remdesivir and chloroquine effectively inhibit the recently emerged novel coronavirus (2019-nCoV) in vitro. *Cell Research*, (January):2019–2021, 2020.

Appendix: Artifact Description/Artifact Evaluation

SUMMARY OF THE EXPERIMENTS REPORTED

All calculations and scalability tests were run with FHI-aims on the new-generation Sunway supercomputer. In order to perform the high-frequency dielectric constants calculation with DFPT, the keyword " DFPT dielectric" needs to be written into control.in; For the DFPT calculations of the polarizability tensor, the keyword " DFPT polar_reduce_memory" for finite systems needs to be presented in the control.in files of FHI-aims. The key algorisms are the calculations of dielectric constants and polarizability listed above.

As described in the paper, we use the case of Silicon Solid and RBD, which can be gotten by https://github.com/mainAIMS/Benchmarks-AIMS.git This data includes example “RBD” data an "Si_core6" data of different cases with folder names like n_100_mini, n_100_rt_mini, et,al.

Then you can install the code in the following way: (1) tar zxvf FHIaims-master.tgz; (2) cd FHIaims-master/src (3) make -f Makefile.sw -j 8 scalapack.mpi

After compiling the FHI-aims, there will be a binary file in the FHIaims-master/bin directory, which is called aims.191127.scalapack.mpi.x

Finally, after finishing the compilation, we can go to the directory which contains the control file (control.in) and the geometry file (geometry.in) , and perform the calculation. We can submit the calculation to the queue using :

bsub -b -m 1 -q q_sw_share -n -share_size 4096 -host_stack 1024 -o output /FHIaims-master/bin/aims.191127.scalapack.mpi.x

Author-Created or Modified Artifacts:

```
Persistent ID:
↪  https://github.com/aims-for-sc21/aims.git
Artifact name: software, the account  username is :
↪  aims-for-sc21

Persistent ID:
↪  https://github.com/mainAIMS/Benchmarks-AIMS.git
Artifact name: input files
```

BASELINE EXPERIMENTAL SETUP, AND MODIFICATIONS MADE FOR THE PAPER

Relevant hardware details: new Sunway: Each processor contains 6 core-groups (CGs), with 65 cores in each CG, and in total 390 cores. Each CG contains one management processing element (MPE), one cluster of computing processing elements (CPEs) and one memory controller (MC). The MPE within each CG is used for computations, management and communication. The CPEs is organized as an 8×8 $mesh$ $(64 cores) and is designed to maximize the aggregated computing power and to minimize the complexity of the micro-architecture.$

Operating systems and versions: Sunway customized Linux with kernel version 3.10.0

Compilers and versions: mpif90

Applications and versions: FHI-aims.191127.scalapack.mpi

Libraries and versions: lapack-v3.8.0, scalapack-v2.0.2