2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)

2020 IEEE 22nd International Conference on High Performance Computing and Communications | 978-1-7281-7649-9/20/$31.00 ©2020 IEEE | DOI: 10.1109/HPCC-SmartCity-DSS50907.2020.00025

An Efficient Multi-GPU Implementation for Linear-Response Time-Dependent Density Functional Theory

Qingcai Jiang†, Lingyun Wan‡, Shizhe Jiao‡, Wei Hu‡, Junshi Chen†* and Hong An†*
†School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
‡Hefei National Laboratory for Physical Sciences at the Microscale, Department of Chemical Physics,
and Synergetic Innovation Center of Quantum Information and Quantum Physics,
University of Science and Technology of China, Hefei, China
Email: †jqc@mail.ustc.edu.cn, ‡{wanly, jsz226}@mail.ustc.edu.cn, ‡whuustc@ustc.edu.cn,
†cjuns@mail.ustc.edu.cn, †han@ustc.edu.cn

Abstract—Nowadays, Kohn-Sham density functional theory (DFT) calculation has drawn more and more attention in chemistry and material science simulations. However, due to the extreme large Hamiltonian matrix needed to be generated during the calculation, when the studied system increases, the cost of calculation becomes unbearable both in ground and excited state electronic structure simulations with large uniform basis. In this paper, we propose a high-performance multi-GPU approach for linear-response time-dependent density functional theory (LR-TDDFT) calculation to compute the excitation energies in molecules and solids with the plane wave basis set under the periodic boundary condition. We carefully design the parallel implementation, calculation steps and data distribution schemes in the naïve CPU implementation to maintain good scalability when the studied system expands, then port the most time-consuming part to multi-GPU platform along with several effective optimization steps. The results show that with dual V100 GPUs, the proposed approach can achieve an average of 6.68x speedup compared with dual 12-core Xeon CPU with bulk silicon systems that comprises thousands of atoms (1,024 atoms).

Index Terms—Linear-response time-dependent density functional theory, GPU, Parallel approach

I. Introduction

Due to its good balance between accuracy and computational efficiency, the time-dependent density functional theory (TDDFT), established by the Runge-Gross theorem [1], which is a self-consistent framework proposed to describe the excited states properties in many systems like molecules and solids, has been widely used in matter physics, quantum chemistry and material science [2]–[4]. Generally speaking, there are two ways to solve the time-dependent Schördinger equation within the framework of TDDFT [5], [6], the most widely used approach is linear-response time-dependent density functional theory (LR-TDDFT), which is often referred directly as TDDFT in literature, solves many-body quantum problems through Fourier transformation of the time-dependent linear response functions. It obtains excitation energies and corresponding oscillation strengths from poles and residues in the complex response function [7], [8].

Within LR-TDDFT framework, the most common way to obtain the excitation energy and corresponding wave functions is to solve the linear response Cassida equation. The most time-consuming part in LR-TDDFT calculation is to explicitly construct the LR-TDDFT Hamiltonian with the complexity of $\mathcal{O}(N_e^5)$ with respect to the number of electrons in the studied system, N_e. So as the system expands, the computation and memory cost of LR-TDDFT calculation in generic CPU platform becomes prohibitively expensive, especially in large uniform basis sets such as plane wave basis set. Therefore, it is still a very challenging work to explore the excited state properties of system with hundreds of atoms with the LR-TDDFT framework.

Fortunately, the situation has been improved thanks to both new algorithm developments and the development of heterogeneous computing architecture like General-Purpose Graphics Processing Unit (GP-GPU). On the algorithmic aspect, we can carry Tamm-Dancoff approximation and use local-density approximation functional to reduce computational cost while maintaining the chemical accuracy at the computational level, as a contrast, traditional LR-TDDFT calculation will carry the computational step with a much larger LR-TDDFT Hamiltonian, which increases the cost of computation, memory and communication. On the hardware aspect, heterogeneous architecture powered by GPU accelerators has become the most widely used architecture among supercomputers, as an example, The Top-2 fastest supercomputer Summit's 95% computing power is provided by the latest NVIDIA Tesla V100 GPU, which has significantly increased the available computing power. Compared with the generic CPU processor, GPU provides much higher floating-point computing efficiency, which can be greatly beneficial for LR-TDDFT calculation.

In this paper, motivated by algorithm and hardware evolvements, we propose a highly efficient multi-GPU implementation of LR-TDDFT, which was designed carefully

*Corresping Author: Hong An and Junshi Chen.

978-1-7281-7649-9/20/$31.00 ©2020 IEEE
DOI 10.1109/HPCC-SmartCity-DSS50907.2020.00025

in mathmatical way and offloads the matrix multiplication, Fast Fourier Transform and several computationally intensive kernels computations from CPUs to commodity GPUs.

We summarize the contribution as follows:

- We design an efficient parallel strategy along with useful data distribution schemes to implement large scale linear-response time-dependent density functional theory.
- We carry out a method for multi-GPU acceleration in LR-TDDFT and propose some resultful optimizations to further reduce the wall clock time.
- We perform extensive experiments to evaluate the performance of the proposed LR-TDDFT calculation on GPUs to prove the effectiveness of our method.

The rest of the manuscript is organized as follows. Section II presents the theoretical algorithm and parallel implementation of LR-TDDFT. Multi-GPU implementation and some other optimizations are shown in section III. The numerical result and corresponding analysis are discussed in Section IV. In Section V, we review the related works. We conclude our work and propose an outlook for our future work in Section VI.

II. Parallel Implementation of LR-TDDFT

In this section, we will introduce the theoretical approach and parallel implementation of LR-TDDFT with the plane wave set on CPU platform in detail. We remark that the scalable implementation of LR-TDDFT is facilitated by the MPI-OpenMP hybrid parallel strategy to reduce the massively high computational cost in constructing and calculating the eigenvalues of LR-TDDFT Hamiltonian.

A. Theoretical Algorithm

LR-TDDFT solves eigenvalue equation of the form

$$HX = \Lambda X \tag{1}$$

Where X represents the coefficient of excitation wavefunction in the Kohn-Sham orbitals and Λ presents the excitation energies. The H is what we call LR-TDDFT Hamiltonian because it has similar form with Hamiltonian in KS-DFT [8].

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

$D\left(i_v i_c, j_v j_c\right) = \left(\varepsilon_{i_c} - \varepsilon_{i_v}\right)\delta_{i_v j_v}\delta_{i_c j_c}$, here δ denotes the Kronecker delta, is an $N_{cv} \times N_{cv}$ ($N_{cv} = N_c \times N_v$ matrix, N_c is the number of conduction orbitals and N_v is the number of valence orbitals) diagonal matrix. These energies and orbitals are typically obtained via the Kohn-Sham density functional theory (KSDFT) [9] calculations. The V_{Hxc} and W_{Hxc} are the Hatree-exchange-correlation integrals [10]

$$\begin{aligned} V_{\mathrm{Hxc}} &= \int \Psi^*_{i_\nu}(\mathbf{r})\Psi_{i_c}(\mathbf{r}) f_{\mathrm{Hxc}}(\mathbf{r},\mathbf{r}')\Psi_{j_\nu}(\mathbf{r}')\Psi^*_{j_c}(\mathbf{r}')\,\mathrm{d}\mathbf{r}\,\mathrm{d}\mathbf{r}' \\ W_{\mathrm{Hxc}} &= \int \Psi^*_{i_\nu}(\mathbf{r})\Psi_{i_c}(\mathbf{r}) f_{\mathrm{Hxc}}(\mathbf{r},\mathbf{r}')\Psi^*_{j_\nu}(\mathbf{r}')\Psi_{j_c}(\mathbf{r}')\,\mathrm{d}\mathbf{r}\,\mathrm{d}\mathbf{r}' \end{aligned} \tag{3}$$

Here f_{Hxc} is the Hatree-exchange-correlation kernel

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

Where $n(\mathbf{r}) = \sum_{i=1}^{N_v} |\Psi_i(\mathbf{r})|^2$ is the electron density and f_{xc} is the exchange-correlation potential in LR-TDDFT calculation.

Thanks to the Tamm-Dancoff approximation (TDA) [11], W_{Hxc} is neglectable and H becomes Hamiltonian matrix with the form

$$H = D + 2V_{\mathrm{Hxc}} \tag{5}$$

So basically we need to get Hartree-exchange-correlation integral V_{Hxc} to construct the LR-TDDFT Hamiltonian, and in discrete cases V_{Hxc} can be constructed as the multiplication of the matrix f_{Hxc} and transposed Khatri-Rao product matrix [12] $P_{vc} = \{\Psi_{i_v}(\mathbf{r})\Psi_{i_c}(\mathbf{r})\}$ with the valence and conduction orbitals ($\Psi_{i_v}(\mathbf{r})$ and $\Psi_{i_c}(\mathbf{r})$) in real space, here N_r denotes the number of grid points for a wavefunction in real space.

$$V_{\mathrm{Hxc}} = P^{\dagger}_{vc} f_{\mathrm{Hxc}} P_{vc} \tag{6}$$

For simplicity and computational scalability, we choose to use the local-density approximation (LDA) functional [13] in the KSDFT and LR-TDDFT calculation. With LDA funtioanal, f_{xc} that denotes exchange-correlation potential, and it is diagonal in real space $\left(\{\mathbf{r_i}\}_{i=1}^{N_r}\right)$, so the exchange-correlation product $f_{xc}P_{vc}$ is more efficiently computed in real space via General Matrix Multiply with a computational complexity of $O(N_r N_v^2 N_c^2) \approx O(N_e^5)$, here N_e denotes the number of electrons in the studied system. We notice that the Hartree potential v_{H} is diagonal in reciprocal space $\left(\{\mathbf{G_i}\}_{i=1}^{N_g}\right)$, here N_g is the grid points in reciprocal space. We transform V_{H} to reciprocal space so $\hat{v}_{\mathrm{H}}\hat{P}_{vc}$ can be computed in a more efficient pattern. $\hat{v}_{\mathrm{H}}\hat{P}_{vc}$ can be calculated by General Matrix Multiply with a computational complexity of $O(N_g N_v^2 N_c^2) \approx O(N_e^5)$ after we use the Fast Fourier Transform (FFT) to carry the transform from real space to reciprocal space. After these steps, we perform FFT to the product of Hartree potential operation between reciprocal and real space to add $v_{\mathrm{H}}P_{vc}$ and $f_{\mathrm{xc}}P_{vc}$ and multiply $P^{\dagger}_{vc}$ to get the result of Hartree-exchange-correlation integral V_{Hxc}.

According to Equation 2, LR-TDDFT Hamiltonian can be constructed by adding V_{Hxc} and D. Right after constructing the LR-TDDFT Hamiltonian H, the rest of the LR-TDDFT calculation is diagonalizing the LR-TDDFT

Hamiltonian H to get the excitation wavefunctions X and energies Λ. The process of LR-TDDFT calculations is presented in Algorithm 1.

B. Parallel Implementation

We implement the LR-TDDFT module to post-process the wavefunctions Ψ and ground-state energies ε_i cacullated by PWDFT (Planewave density functional theory) [14], which is a submodule of DGDFT (Discontinuous Galerkin Density Functional Theory) [15], [16]. DGDFT is a massively parallel software package to perform large-scale Kohn-Sham density functional theory (DFT) calculations efficiently and PWDFT is a self-contained module for performing conventional standard plane-wave based electronic structure calculations. To maintain the high computational efficiency at a large scale, we design a MPI-OpenMP hybrid parallelization strategy in LR-TDDFT implementation. In specific, we use MPI to handle different types of data and task distribution schemes and we use OpenMP to further discover the parallelism of each MPI task so it can be fully scalable on modern supercomputers.

As we can see from figure 1, there are two main data distribution schemes for the wavefunctions in the LR-TDDFT implementation. The first one is column block partition, each column of wavefunction are stored to one MPI task based on its column index. This data distribution scheme is highly efficient to apply Hatree operator since different MPI tasks are able to perform FFTs independently in reciprocal space. The second one is row block partition, in this scheme the data is distributed according to the partition of MPI tasks. This distribution scheme facilitates the evaluation of matrix-matrix multiplication (GEMM). In LR-TDDFT the wavefunctions Ψ are mostly distributed in the column block partition to present superposition of a set of eigenwave functions in the system. The conversion from column block partition to row block partition is perform via MPI_Alltoall.

Algorithm 1 The pseudocode for carrying the LR-TDDFT calculations.

Input:
Ground-state energies ϵ_i
Wavefunctions $\Psi_\mu(\mathbf{r})$ and $\Psi_\nu(\mathbf{r})$ distributed according to the column block index.

1: for each MPI task in LR-TDDFT calculation do
2: Initialize $P_{vc}(r) = \{\Psi_\mu(r)\Psi_\nu(r)\}$ in real space $\sim \mathcal{O}(N_v N_c N_r)$
3: Carry out MPI_Alltoall to wavefunctions Ψ to transfer data distribution scheme from row block partition to column block partition
4: transfer $P_{vc}(r)$ into reciprocal space (fftR2C) $\sim \mathcal{O}(N_r \log N_r N_v N_c)$
5: Apply the Hartree potential in reciprocal space and transfer into real space $v_{\mathrm{H}} P_{vc} \sim \mathcal{O}(N_r \log N_r N_v N_c + N_c N_v N_r)$
6: Carry out MPI_Alltoall to wavefunctions Ψ to transfer data distribution scheme from column block partition to row block partition
7: Compute the Hartree-exchange-correlation integrals V_{Hxc} in real space (GEMM) $\sim \mathcal{O}(N_r N_v^2 N_c^2)$
8: summarize V_{Hxc} within all MPI tasks by MPI_Allreduce
9: end for
10: Obtain LR-TDDFT Hamiltonian by computing the difference of Kohn-Sham energy eigenvalues
11: Diagonalize the LR-TDDFT Hamiltonian H

Output: Excited-state energies $\{\lambda_i\}$ and wavefunctions $\{x_{ij}\}$

III. Multi-GPU Acceleration

To compute the excitation energies in molecules and solids with plane wave basis set, it usually takes 80% of wall clock time to apply the Hatree potential in reciprocal space and compute the Hatree-exchange-correlation integrals in real space with our Naïve CPU implementation. To be exact, the most time consuming part consists of operations such as Fast Fourier Transform (FFT), General Matrix Multiply (GEMM), matrix diagonalization (SYEVD) and some MPI communications such as MPI_Allreduce. These matrix operations can be done via the level 3 Basic Linear Algebra Subprograms (BLAS), Linear Algebra PACKage (LAPACK) and Fastest Fourier Transform in the West (FFTW) libraries in the CPU implementation. To fully accelerate the time-intensive part of LR-TDDFT, we port these kernels to GPUs with either GPU-accelerated or CUDA custom kernels. We also carry a better data partition way to overlap the MPI communications and GPU computation in order to fully explore the parallel potential of the heterogeneous architecture.

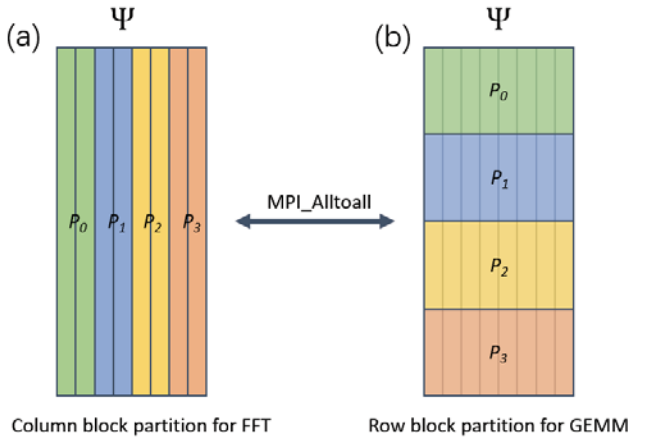

Fig. 1. Parallel data and task distribution schemes of LR-TDDFT. (a) Band parallelization with column block partition (for FFT) and (b) grid parallelization with row block partition (for GEMM). Illustration of the parallel scheme with wavefunction number $N_{cv} = 8$ and 4 computation cores.

In order to carry out efficient acceleration on multi-GPU platform, we perform a serial of optimization steps.

A. Point-to-point Replacement

The first step of porting LR-TDDFT to GPU is to replace the most computationally expensive part kernels like FFT and GEMM by using cuBLAS, cuFFT and CUDA custom kernels to accelerate the computation of Algorithm 1. In our implementation, the CUDA custom kernels are written to fill the gaps between the cuFFT functions as a damping coefficient to accelerate the process of convergence. In our CPU implementation, the data distribution scheme is perfect for multi-GPU parallelization because each MPI task holds all the data needed to perform numerical operations, so there is no GPU-GPU data transfer between calculation steps. After mapping these kernels to GPU, the CPUs only perform a spot of computation steps, and most of the CPU time are used to carry MPI communication, and the data copy between CPU and GPU is necessary before and after each GPU calculation operation.

B. Overlap of computation and communication.

In principle, CUDA-aware MPI communication can naturally overlap with the GPU computation. But as we can see from Algorithm 1, right after computing the Hartree-exchange-correlation integrals via General Matrix Multiply (GEMM), we perform MPI_Allreduce to gather V_{Hxc} in all MPI tasks. Due to data dependence, MPI_Allreduce must wait for GEMM to finish computation, thus the overlapping of computation and communication is disrupted. When the system size increases, although GPUs can handle GEMM in a very efficient way, both GEMM and MPI_Allreduce will introduce much time cost. To fully accelerate LR-TDDFT, we made an attempt to achieve the overlap of computation and communication.

First of all, after analyzing the data partition of LR-TDDFT, we find that to calculate the difference between Kohn-Sham energy eigenvalues, not all MPI tasks need to store the whole V_{Hxc} matrix. So after GEMM, we change the way of data partition, as we can see from Figure 2, then we get the result of GEMM, each MPI task only needs to store part of the V_{Hxc} matrix.

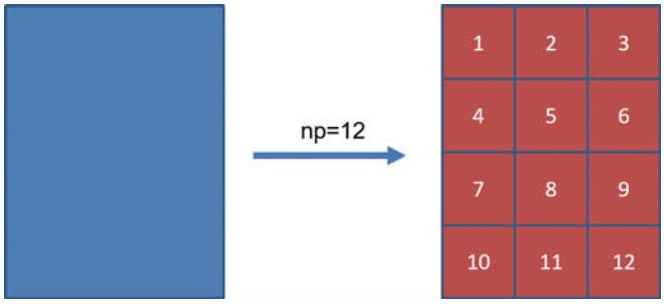

Fig. 2. Data partition optimization of V_{Hxc}

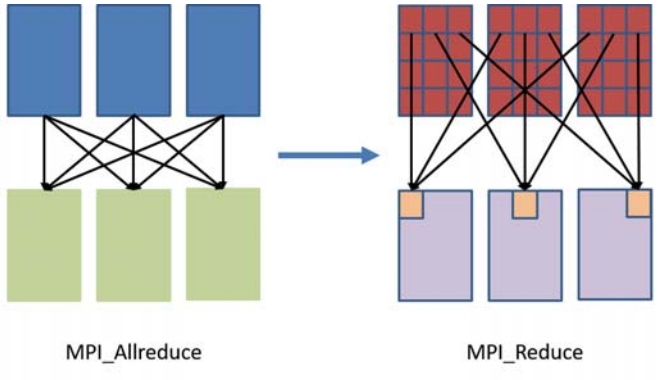

Fig. 3. Data reduction optimization, take the first row of Matrix V_{Hxc} as example

The above attempts bring about two merits. First, this data partition way can reduce memory usage for the number of MPI processes times. Second, we will not need to perform MPI_Allreduce to gather the whole V_{Hxc} matrix, instead, we use MPI_Reduce to transfer part of V_{Hxc} matrix to each MPI task based on the index.

Owing to these attempts we eliminate part of the data dependencies, to be detailed, we can split the matrix into small blocks and perform GEMM manually to these small blocks. The basic flow of GEMM and Reduce is shown in Figure 4 and ,the pipeline of GEMM and reduce after our optimization are shown in Figure 5. Once each block gets the result, we can immediately reduce the block matrix to each MPI task by MPI_Reduce. Afterward we will get the whole V_{Hxc} but distributed in each MPI task.

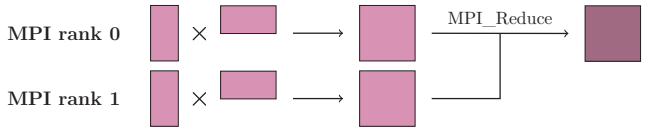

Fig. 4. The basic implementation of GEMM and Reduce, take MPI task = 2 as example

C. Mixed-precision computation

Since the element V_{Hxc} in the Hamiltonian matrix of the LR-TDDFT equation is much smaller than D, the excitation energy of LR-TDDFT is significantly affected by the energy level difference, in this way we can relax the accuracy in the construction process of V_{Hxc} to a certain extent. To further optimize the code while remaining computational accuracy, we test different levels of mixed precision. As a result, we find an accurate and fast prescription that uses single precision in GEMM and FFT operations but uses double precision in initializing $P_{vc}(r)$ in real space and diagonalizing the LR-TDDFT Hamiltonian H. To be exact, the Ψ matrix held by each MPI task in row block partition is converted from double precision to single

precision before the GEMM and FFT, when carrying the other numerical calculations, we transfer the output matrix to double precision to maintain accuracy.

We compare the mixed-precision implementation with the double precision model by using a typical configuration of a silicon crystal system composed of 1024 atoms and with $N_{vc} = 128 \times 128 = 16384$, and observe a deviation of root mean square 0.29eV/molecule in the energy prediction. Since these deviations are much less than the error caused by local-density approximation, so the mixed precision implementation is of satisfactory accuracy. In terms of speed and GPU memory requirement, the kernel with mixed precision is ∼1.7 times faster than and consumes 50% less memory than the double precision version. After testing a lot of other mixed precision plans, we remark that although a fully-float implementation is more power efficient on the NVIDIA V100 GPU, our tests show that, due to the limited representation range with 32 binary bits, the corresponding attempt cannot preserve the required accuracy of the excited energy and wavefunctions. Therefore, at this moment, we cannot take advantage of the merits of the fully-float implementation.

D. Memory access optimization

In the baseline GPU implementation, matrix fxc and vector $Eigenvalue$ are stored in the GPU global memory for the penultimate part of the computation to calculate $fxc+ = Eigenvalue_j - Eigenvalue_k$. However, GPU global memory has high latency and low bandwidth compared to the shared memory inside each GPU streaming multiprocessor (SM). We remark that fxc is calculated plus the element-wise product of $Eigenvalue_j - Eigenvalue_k$. To improve the performance, we use the low-latency on-chip shared memory to replace the GPU global memory to store fxc and $Eigenvalue$. Then, when calculating fxc, the algorithm accesses the shared memory that is much faster than accessing GPU global memory.

E. Single precision MPI

After the above optimizations of LR-TDDFT, we found that as the number of MPI task grows, the wall-clock time of LR-TDDFT is dominated by 2 steps of MPI_Alltoall operation. This result is quite intuitive because MPI_Alltoall sends data from all to all processes, meanwhile, the data is very large as the size of input increases. To reduce the communication time of the data transform between schemes in Figure 1, we use the single precision format for sending and receiving the wavefunctions, which reduces the communication throughput by half. We also mention that the single precision format is only used in the MPI_Alltoall communication, so wavefunctions will be converted back to the double precision format for computation. We find that this optimization leads to negligible deviation in the accuracy of the result in the LR-TDDFT, in our experiment, the error of the total energy is as little as 10^{-7}eV/molecule for a silicon system with 1024 atoms and $N_{vc} = 128 \times 128 = 16384$, compared to the double precision implementation.

Figure 6 shows the reduction of the computational time associated with different steps of optimization. The testing system is a 1024 silicon atoms system shown in Figure 7, which will be discussed in section V. The naïve CPU version of LR-TDDFT uses dual Intel Xeon E5-2695 12-core sockets and in our tests we use 6 CPU core bound to each MPI task. The GPU version uses dual V100 GPU in NVIDIA DGX-1 server [17], and is around 6.7 times faster than the CPU version in terms of LR-TDDFT calculation. We find that step 1 (point-to-point replacemtent) leads to most of the performance improvement from CPUs to GPUs, and this step is responsible for most of the implementation efforts as well.

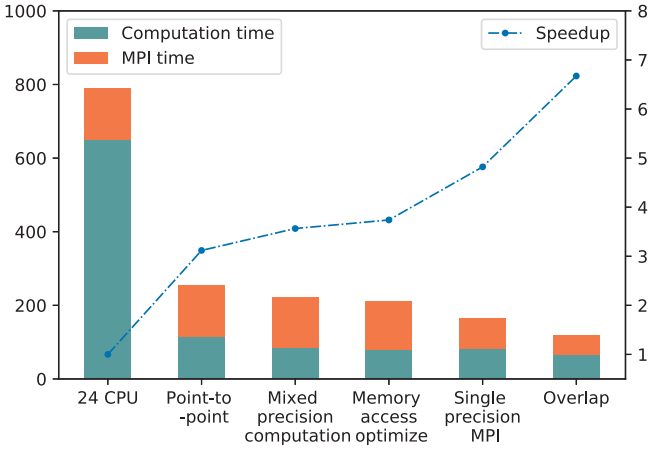

Fig. 6. Wall clock time for LR-TDDFT calculation with different steps of optimizations for a system with 1024 silicon atoms. The CPU version uses 24 CPU cores, and the GPU version uses 2 GPUs

IV. Numerical Result and Analysis

In this section, we report the numerical results, including the speedup, strong and weal scaling performance of the LRTDDFT code with several sizes of bulk silicon systems. Different size of testing systems are generated to faithfully indicate the performance of LR-TDDFT with GPU accelerated, including silicon systems with 64, 256, 512, 1024, 2048 atoms which will be later remarked as Si64, Si256, Si512, Si1024 and Si2048, respectively. In our tests, we perform the number of valence and conduction orbitals $N_v = N_c = 180$, thus the height and length of the LR-TDDFT Hamiltonian are both $N_{\mathrm{cv}} = 180 \times 180 = 32400$. We perform all numerical tests on NVIDIA DGX-1 server shown in Figure 8. DGX-1 server has dual 20-Core Intel Xeon E5-2698 CPU sockets running at 2.2 GHz and 8 Tesla V100 GPU connected via PCIe 3.0. V100 GPUs are connected via NVLink with a bandwidth of 20GB/s. The Tesla V100 GPU has 16 GB device memory and 5120 CUDA cores running at 1.38 GHz, which provide

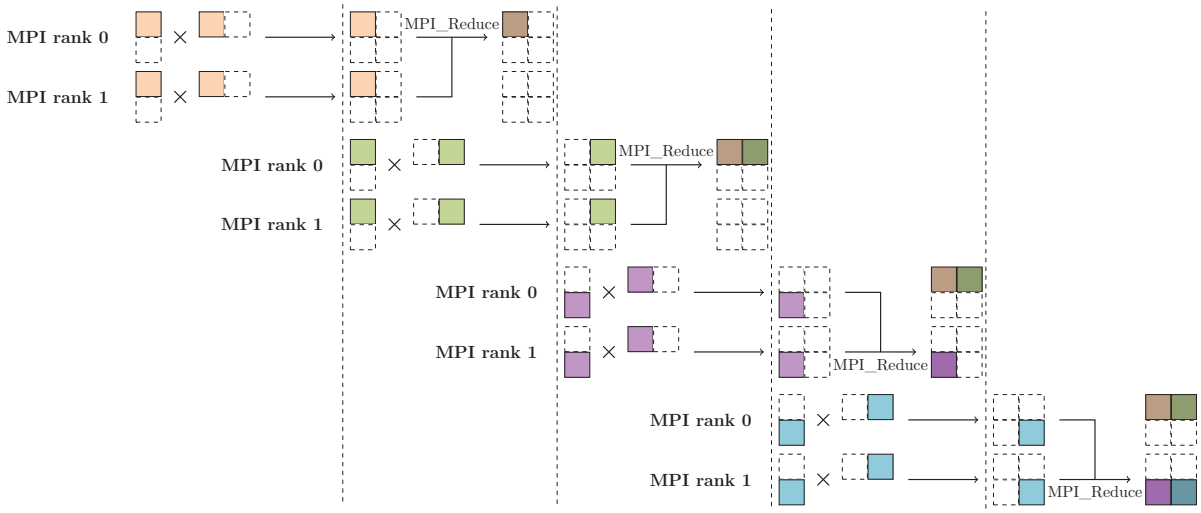

Fig. 5. Overlap of computation and communication, this figure shows the pipeline of GEMM and Reduce, we split the matrices mannualy to calculate several small block matrices. Take MPI task = 2 as example

a theoretical peak performance of 7.8TFLOPS double precision operations. The operating system of the server is 64-bit Red Hat Enterprise Linux and the system has in total 512GB DRAM memory.

In this paper, the CPU algorithm and GPU algorithm are running on C++ and CUDA 10.0, respectively. Which we test GPU versions of LR-TDDFT, each MPI task is bound to an individual GPU and each experiment is executed four times and we report the average results.

A. Speedup

At first, we perform experiments with several sizes of silicon systems to make sure our implementation works well on systems of different sizes. Three versions of the algorithm are executed for comparison: the CPU version, the unoptimized GPU version, and the optimized GPU version. The unoptimized and optimized GPU version stand for the point-to-point approach and the final implementation, respectively. The CPU version uses 20 CPU core and 5 MPI tasks, so each MPI task is bound to 4 CPU

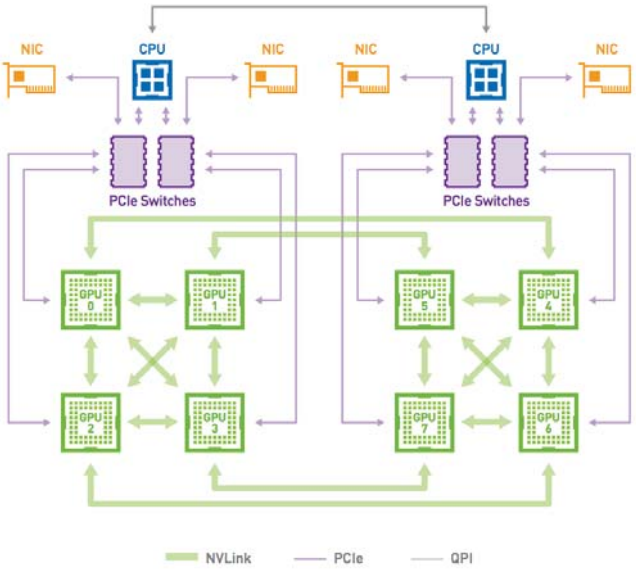

Fig. 8. The architecture of NVIDIA DGX-1.

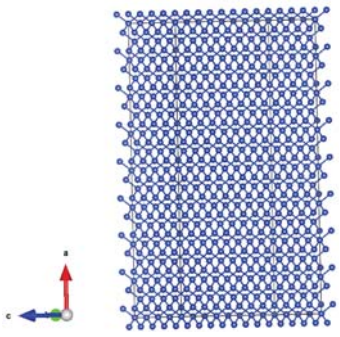

Fig. 7. Bulk silicon systems with 1024 atoms

core and the data dispatch and multithread execution are provided by OpenMP. Figure 9 shows the running time and speedups of LR-TDDFT calculation running on the Tesla V100 GPUs and CPUs, respectively. The optimized GPU algorithm achieves an average of 5.75× and up to 6.94× speedups versus CPU version. In contrast, the unoptimized GPU algorithm achieves an average of 3.01× and up to 3.20× speedups CPU version. The results demonstrate the effectiveness and universality of the optimization strategies in Section III. The figure also shows that the speedups keep increasing with the growing of system size. However, the

maximum system size that can be tested on the GPU is Si2048 due to limited GPU memory.

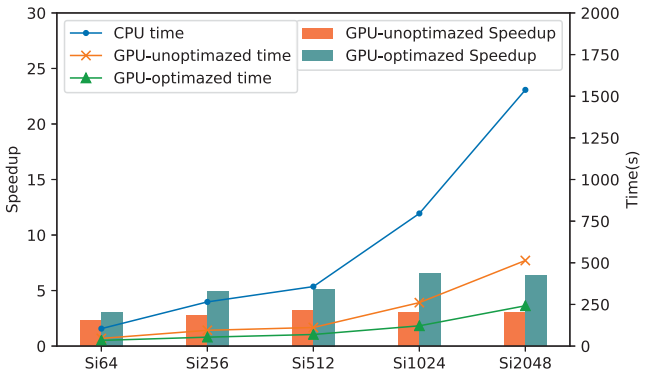

Fig. 9. Running time and speedups of difference sizes of testing systems on two platforms. GPU-unoptimized indicates our naïve implement and GPU-optimized represents our final version after 4 steps of optimizations

B. Strong Scaling

The most significant concern for the LR-TDDFT calculations is the strong scalability on heterogeneous multi-GPU architectures, which reflects how much does the growth of hardware resources takes effect for a determined system as adding the numbers of GPUs. We test the GPU-optimized version of LR-TDDFT on bulk silicon systems with 1024 atoms and we evaluate strong scalability by parallel efficiency defined in Equation 7, and the speedup is compared with wall clock time in single GPU.

$$\text{Efficiency} = \frac{\text{speedup}}{Number\ of\ GPU} \tag{7}$$

As we can see from Figure 10, parallel efficiency goes down when the number of GPU increases. The main reason for this phenomenon is that since we distribute matrices into each MPI task equably which bounds to a GPU and the size of each matrix performed GEMM and FFT becomes smaller when we use more GPUs, so GPU cannot effectively use its computing power. But parallel efficiency is still above 55%, which is quite acceptable since in the calculation it exists many times of global communication.

C. Weak Scaling

Another important criterion for the LR-TDDFT calculations is the weak scalability, which reflects the parallel performance for a scaled problem size along with a fixed number of GPU. Different sizes of silicon systems are tested on 2 V100 GPUs between Unoptimized and Optimized GPU versions. Table I show that compared with Si512 system, Si1024 and Si2048 achieve almost linear scaling in optimized GPU version, which demonstrates the scalability of our implementation.

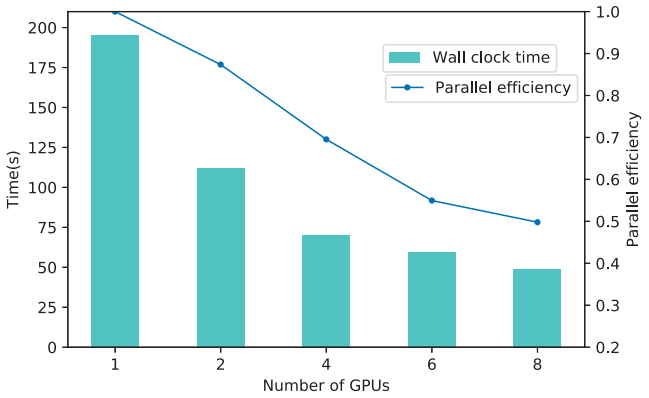

Fig. 10. Strong scaling: the total time and parallel efficiency of GPU-optimized version using different number of GPUs.

TABLE I
Wall clock time of different scale of testing system with 2 GPU

Testing system	GPU Time	
	Unoptimized	Optimized
Si64	45.98	34.13
Si256	94.68	53.51
Si512	111.6	69.53
Si1024	260.95	121.68
Si2048	503.66	221.9

V. Related Work

In ground state electronic structure calculations, various highly efficient HPC KS-DFT software packages use GPU to reduce computation time such as ABINIT [18], BigDFT [19], Octopus [2], PWmat [20], [21], Quantum ESPRESSO [22], DGDFT [15], [23] and VASP [24], [25], which are beneficial to take full advantage of the massive parallelism available on modern heterogeneous systems. But only several HPC software packages for excited state electronic structure calculations have been developed, such as NWChem [26], BerkeleyGW [27] and QChem [28], due to such ultrahigh computation and memory cost in the excited state electronic structure calculations, especially with plane wave basis set. In particular, Jia et al. [29] accelerate hybrid functional rt-TDDFT calculations, which is another form of TDDFT, using the parallel transport gauge formalism by GPU on Summit, their implementation can efficiently scale to 786 GPUs for a large system with 1536 silicon atoms.

VI. Conclusion and Outlook

In this paper, we propose an Efficient multi-GPU Implementation for linear-response time-dependent density functional theory calculation to compute the excitation energies in molecules and solids with the plane wave basis set. We carefully design the mathematical approximation model to reduce the computational cost while retaining chemical accuracy. In our naïve CPU design, we perform different data partition and task distribution schemes to

ensure the flexibility and efficiency of the computation process. Meanwhile, hybrid MPI-OpenMP programming method is used to achieve high scalability in modern multi-core computing system. The most challenging part of LR-TDDFT calculation is to generate the LR-TDDFT Hamiltonian, which usually occupies 80% of the whole wall clock time, so we implement a multi-GPU version by port the most time consuming part to multi-GPU and attempt several optimizations to fully tap the potential of modern heterogeneous GPU system. We perform numerical testing among different size of systems, and GPU version achieves an average of 5.75× and up to 6.94× speedups versus CPU version, and both strong and weak scaling keep at a high level.

Looking forward to our future work, there will still remain a lot of challengs to further explore the excited state properties of molecules and solids with modern heterogeneous computing architecture. On one hand, we will afterward seek the method to effectively use the massively computing power in Sunway TaihuLight supercomputer [30] within LR-TDDFT calculation. On the other hand, the limitation of LR-TDDFT is that the accuracy cann't meet precise demand of simulated spectrum, so we try to boost the accuracy of excited state properties by using more elaborate mathematical method like GW/BSE [31].

Acknowledgment

We are thankful to the reviewers for evaluating this study and providing valuable feedback. This work is supported by the National Key Research and Development Program of China (Grants No.2018YFB0204100).

References

[1] E. Runge and E. K. U. Gross, "Density-functional theory for time-dependent systems," Phys. Rev. Lett., vol. 52, pp. 997–1000, Mar 1984. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.52.997

[2] X. Andrade, J. Alberdi-Rodriguez, D. A. Strubbe, M. J. Oliveira, F. Nogueira, A. Castro, J. Muguerza, A. Arruabarrena, S. G. Louie, A. Aspuru-Guzik et al., "Time-dependent density-functional theory in massively parallel computer architectures: the octopus project," Journal of Physics: Condensed Matter, vol. 24, no. 23, p. 233202, 2012.

[3] C. A. Ullrich, Time-dependent density-functional theory: concepts and applications. OUP Oxford, 2011.

[4] K. Yabana and G. Bertsch, "Time-dependent local-density approximation in real time," Physical Review B, vol. 54, no. 7, p. 4484, 1996.

[5] T. L. Beck, "Real-space mesh techniques in density-functional theory," Reviews of Modern Physics, vol. 72, no. 4, p. 1041, 2000.

[6] G. Onida, L. Reining, and A. Rubio, "Electronic excitations: density-functional versus many-body green's-function approaches," Rev. Mod. Phys., vol. 74, pp. 601–659, Jun 2002. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.74.601

[7] M. E. Casida and D. Chong, "Recent advances in density functional methods," Computational Chemistry: Reviews of Current Trends, 1995.

[8] R. Sternheimer, "Electronic polarizabilities of ions from the hartree-fock wave functions," Physical Review, vol. 96, no. 4, p. 951, 1954.

[9] P. Hohenberg and W. Kohn, "Inhomogeneous electron gas," Physical review, vol. 136, no. 3B, p. B864, 1964.

[10] W. Kohn and L. J. Sham, "Self-consistent equations including exchange and correlation effects," Physical review, vol. 140, no. 4A, p. A1133, 1965.

[11] A. L. Fetter and J. D. Walecka, "Quantum theory of many-particle systems," qtmp, 1971.

[12] C. Khatri and C. R. Rao, "Solutions to some functional equations and their applications to characterization of probability distributions," Sankhyā: The Indian Journal of Statistics, Series A, pp. 167–180, 1968.

[13] S. Goedecker, M. Teter, and J. Hutter, "Separable dual-space gaussian pseudopotentials," Physical Review B, vol. 54, no. 3, p. 1703, 1996.

[14] W. Hu, L. Lin, A. S. Banerjee, E. Vecharynski, and C. Yang, "Adaptively compressed exchange operator for large-scale hybrid density functional calculations with applications to the adsorption of water on silicene," Journal of chemical theory and computation, vol. 13, no. 3, pp. 1188–1198, 2017.

[15] W. Hu, L. Lin, and C. Yang, "Dgdft: A massively parallel method for large scale density functional theory calculations," The Journal of chemical physics, vol. 143, no. 12, p. 124110, 2015.

[16] L. Lin, J. Lu, L. Ying, and E. Weinan, "Adaptive local basis set for kohn–sham density functional theory in a discontinuous galerkin framework i: Total energy calculation," Journal of Computational Physics, vol. 231, no. 4, pp. 2140–2154, 2012.

[17] Nvidia dgx-1 with teslav100 system architecture. [Online]. Available: https://images.nvidia.com/content/pdf/dgx1-v100-system-architecture-whitepaper.pdf

[18] X. Gonze, F. Jollet, F. A. Araujo, D. Adams, B. Amadon, T. Applencourt, C. Audouze, J.-M. Beuken, J. Bieder, A. Bokhanchuk et al., "Recent developments in the abinit software package," Computer Physics Communications, vol. 205, pp. 106–131, 2016.

[19] L. E. Ratcliff, A. Degomme, J. A. Flores-Livas, S. Goedecker, and L. Genovese, "Affordable and accurate large-scale hybrid-functional calculations on gpu-accelerated supercomputers," Journal of Physics: Condensed Matter, vol. 30, no. 9, p. 095901, 2018.

[20] W. Jia, Z. Cao, L. Wang, J. Fu, X. Chi, W. Gao, and L.-W. Wang, "The analysis of a plane wave pseudopotential density functional theory code on a gpu machine," Computer Physics Communications, vol. 184, no. 1, pp. 9–18, 2013.

[21] W. Jia, J. Fu, Z. Cao, L. Wang, X. Chi, W. Gao, and L.-W. Wang, "Fast plane wave density functional theory molecular dynamics calculations on multi-gpu machines," Journal of Computational Physics, vol. 251, pp. 102–115, 2013.

[22] J. Romero, E. Phillips, G. Ruetsch, M. Fatica, F. Spiga, and P. Giannozzi, "A performance study of quantum espresso's pwscf code on multi-core and gpu systems," in International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems. Springer, 2017, pp. 67–87.

[23] W. Hu, X. Qin, Q. Jiang, J. Chen, H. An, W. Jia, F. Li, X. Liu, D. Chen, F. Liu et al., "High performance computing of dgdft for tens of thousands of atoms using millions of cores on sunway taihulight," Science Bulletin, 2020.

[24] M. Hacene, A. Anciaux-Sedrakian, X. Rozanska, D. Klahr, T. Guignon, and P. Fleurat-Lessard, "Accelerating vasp electronic structure calculations using graphic processing units," Journal of computational chemistry, vol. 33, no. 32, pp. 2581–2589, 2012.

[25] M. Hutchinson and M. Widom, "Vasp on a gpu: Application to exact-exchange calculations of the stability of elemental boron," Computer Physics Communications, vol. 183, no. 7, pp. 1422–1426, 2012.

[26] M. Valiev, E. J. Bylaska, N. Govind, K. Kowalski, T. P. Straatsma, H. J. Van Dam, D. Wang, J. Nieplocha, E. Apra, T. L. Windus et al., "Nwchem: A comprehensive and scalable open-source solution for large scale molecular simulations," Computer Physics Communications, vol. 181, no. 9, pp. 1477–1489, 2010.

[27] J. Deslippe, G. Samsonidze, D. A. Strubbe, M. Jain, M. L. Cohen, and S. G. Louie, "Berkeleygw: A massively parallel computer package for the calculation of the quasiparticle and optical properties of materials and nanostructures," Computer Physics Communications, vol. 183, no. 6, pp. 1269–1289, 2012.

[28] Y. Shao, Z. Gan, E. Epifanovsky, A. T. Gilbert, M. Wormit, J. Kussmann, A. W. Lange, A. Behn, J. Deng, X. Feng et al., "Advances in molecular quantum chemistry contained in the q-chem 4 program package," Molecular Physics, vol. 113, no. 2, pp. 184–215, 2015.
[29] W. Jia, L.-W. Wang, and L. Lin, "Parallel transport time-dependent density functional theory calculations with hybrid functional on summit," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2019, pp. 1–23.
[30] H. Fu, J. Liao, J. Yang, L. Wang, Z. Song, X. Huang, C. Yang, W. Xue, F. Liu, F. Qiao et al., "The sunway taihulight supercomputer: system and applications," Science China Information Sciences, vol. 59, no. 7, p. 072001, 2016.
[31] G. Strinati, "Application of the green's functions method to the study of the optical properties of semiconductors," La Rivista del Nuovo Cimento (1978-1999), vol. 11, no. 12, pp. 1–86, 1988.