Computer Physics Communications 211 (2017) 54–60

Contents lists available at ScienceDirect

Computer Physics Communications

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

Parallel 3-dim fast Fourier transforms with load balancing of the plane waves

Xingyu Gao [a,b,c], Zeyao Mo [a,b,c], Jun Fang [b,c], Haifeng Song [a,b,c], Han Wang [b,c,*]

[a] *Laboratory of Computational Physics, Huayuan Road 6, Beijing 100088, PR China*
[b] *Institute of Applied Physics and Computational Mathematics, Fenghao East Road 2, Beijing 100094, PR China*
[c] *CAEP Software Center for High Performance Numerical Simulation, Huayuan Road 6, Beijing 100088, PR China*

ARTICLE INFO

Article history:
Received 29 January 2016
Received in revised form
23 May 2016
Accepted 2 July 2016
Available online 7 July 2016

Keywords:
First-principles calculation
Kohn–Sham equation
Plane wave
FFT
Load balancing

ABSTRACT

The plane wave method is most widely used for solving the Kohn–Sham equations in first-principles materials science computations. In this procedure, the three-dimensional (3-dim) trial wave functions' fast Fourier transform (FFT) is a regular operation and one of the most demanding algorithms in terms of the scalability on a parallel machine. We propose a new partitioning algorithm for the 3-dim FFT grid to accomplish the trade-off between the communication overhead and load balancing of the plane waves. It is shown by qualitative analysis and numerical results that our approach could scale the plane wave first-principles calculations up to more nodes.

1. Introduction

In the context of Density Functional Theory (DFT), solving the Kohn–Sham equation is the most time-consuming part of the first-principles materials science computations [1–3]. The plane wave method, which is a widely used numerical approach [4], could lead to a large-scale dense algebraic eigenvalue problem. This problem is usually solved by iterative diagonalization methods such as Davidson's [5], RMM-DIIS [3], LOBPCG [6], Chebyshev polynomial filtering subspace iteration [7], etc. The elementary operation of the iteration methods is the matrix–vector multiplication. Since the large-scale dense matrix is not suitable for explicit assembly, we realize the matrix–vector multiplication by applying the Hamiltonian operator on trial wave functions. The local term of the effective potential is one part of the Hamiltonian operator. In order to compute its action in a lower time complexity, we perform 3-dim FFT twice on one trial wave function in each matrix–vector multiplication.

There are three features to make the trial wave function's FFT one of the most demanding algorithms to scale on a parallel machine. The first is the moderate sized FFT grid rather than a large one. The ratio of computation to communication of the parallel 3-dim FFT is of order $\log N$ where N, the single dimension of the FFT grid, is usually $\mathcal{O}(10^2)$ in most first-principles calculations of bulk materials. The second is the accumulated communication overhead led by many execution times corresponding to a number of wave functions. Thousands of FFTs may be executed at each step of iterative diagonalization. The third is the all-to-all communication required by the data transposes. This could limit the parallel scaling due to the large number of small messages in the network resulting in competition as well as latency issues.

It has already been recognized that making fewer and larger messages can speed up parallel trial wave functions' FFTs. The hybrid OpenMP/MPI implementation [8,9] can lead to fewer and larger messages compared to a pure MPI version. And a blocked version [9] performs a number of trial wave functions' FFTs at the same time to aggregate the message sizes and reduce the latency problem.

In first-principles calculations, we should consider not only the parallel scaling of trial wave functions' FFTs, but also the load balancing of intensive computations on the plane waves that expand the wave functions. The workload of these computations are inhomogeneously distributed on a standard 3-dim FFT grid. Thus a greedy algorithm is usually used to optimize the load balancing. However, this algorithm results in global all-to-all communications across all the processors, thus the latency overhead would grow in proportion to the number of processors

* Corresponding author at: Institute of Applied Physics and Computational Mathematics, Fenghao East Road 2, Beijing 100094, PR China.
E-mail address: wang_han@iapcm.ac.cn (H. Wang).

http://dx.doi.org/10.1016/j.cpc.2016.07.001
0010-4655/© 2016 Elsevier B.V. All rights reserved.

and might contribute substantially to the total simulation time. Haynes et al. [10] present a partitioning approach for the 3-dim FFT grid that minimizes the latency cost. Their method depends critically on the Danielson–Lanczos Lemma [11] and requires a particular data distribution, which limits the possibilities to improve the load balancing of the plane waves.

In this paper, we propose a new partitioning method for the 3-dim FFT grid, with which we need independent local all-to-all communications for each data transpose rather than one global all-to-all communication. With this communication pattern preserved, we develop the method to improve the load balancing by adjusting the data distribution among working processors. By numerical examples, we show that although its load balancing is not as perfect as that of the greedy algorithm, the new approach can be more favorable for parallel scaling by making the fewer and larger messages. Hence, we are allowed to accomplish the trade-off between the load balancing of the plane waves and communication overhead in the trial wave functions' FFTs. And such a trade-off could scale the plane wave first-principles calculations up to more nodes. With the proposed partitioning method, we design a compact parallel 3-dim FFT to reduce the amount of calculations and passing messages without loss of accuracy.

The rest of this paper is organized as follows. In Section 2, we explain the elemental role of trial wave functions' FFTs in the plane wave method. In Section 3, we introduce the greedy algorithm for load balancing of the plane waves and analyze the resulting communication cost. In Section 4, we describe the new partitioning algorithms and implementations. In Section 5, we show the numerical results. The last section gives concluding remarks.

2. The role of trial wave functions' FFT

In this section, we explain the elemental role of trial wave functions' FFTs in solving the Kohn–Sham equation using a plane wave basis set.

In the pseudopotential (norm-conserving [12] or ultrasoft [13] pseudopotential) setting or the projector augmented wave (PAW) [14,15] approach, the pseudo wave function $\tilde{\Psi}_i$ satisfies the Kohn–Sham equation which looks like

$$\left(-\frac{1}{2}\Delta + V_{\text{loc}} + V_{\text{nl}}\right)\tilde{\Psi}_i = \epsilon_i S \tilde{\Psi}_i, \tag{1}$$

where $-\frac{1}{2}\Delta$ is the kinetic energy operator, V_{loc} the local potential, V_{nl} the nonlocal term, and S the overlapping operator. In the case of the norm-conserving pseudopotential, S could simply be interpreted as the identity operator. In this manuscript, we refer to the pseudo wave function simply as the wave function.

We use always the periodic boundary condition and expand the wave function in plane waves:

$$\tilde{\Psi}_{n\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{G}} \tilde{\Psi}_{n\mathbf{k}}(\mathbf{G})\, e^{-\imath(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}}, \tag{2}$$

where the $\mathbf{k}$'s are vectors sampling the first Brillouin zone, n is an index of the energy level with given $\mathbf{k}$, and $\mathbf{G}$'s are the reciprocal lattice vectors. The expansion (2) only includes the plane waves satisfying

$$|\mathbf{k}+\mathbf{G}| < \sqrt{2E_{\text{cut}}} \equiv G_{\text{cut}}. \tag{3}$$

In the plane wave method, the Hamiltonian matrix is not assembled explicitly. Instead, iterative diagonalization techniques are employed together with the implicit matrix–vector multiplication that is realized as the action of the Hamiltonian operator on the trial wave functions. It is noticed that the local potential is diagonal in the real space. In order to obtain efficiently the action of the

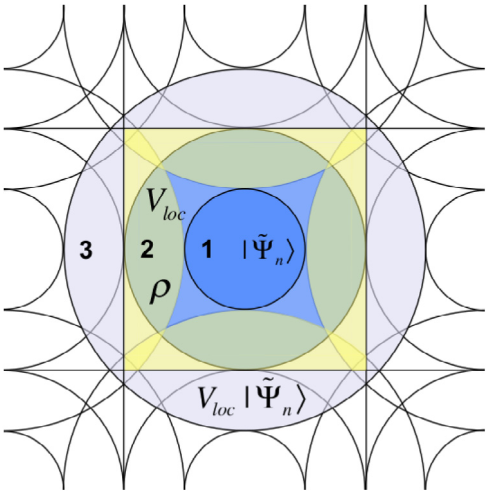


Fig. 1. A two dimensional sketch of the wrap-around errors in the reciprocal space. The wave function $|\Psi\rangle$ is sampled within a sphere with radius G_{cut} (the innermost circle 1). The charge density ρ and the local potential V_{loc} are defined inside a sphere with radius $2G_{\text{cut}}$ (circle 2). We would require a sphere with radius $3G_{\text{cut}}$ to accurately estimate the operation of the local potential on the trial wave function. If we apply a smaller FFT grid with only $2G_{\text{cut}}$, the artificial wrap-around error between $2G_{\text{cut}}$ and $3G_{\text{cut}}$ would occur and be folded back into circle 2 due to the periodicity. Hence, it is sufficient to approximate the wave functions correctly in circle 1.

local potential on the trial wave function, we should first transform $\tilde{\Psi}_{n\mathbf{k}}(\mathbf{G})$ to the real space representation $\tilde{\Psi}_{n\mathbf{k}}(\mathbf{r})$ by one FFT, multiply it with the local potential term, and then transform the product back to the reciprocal space. Consequently, two 3-dim FFTs are required by each action on a trial wave function.

3. The load balancing issue and the greedy algorithm

3.1. The load balancing issue

As mentioned in the previous section, the plane waves are truncated at cut-off radius G_{cut}. Since charge density ρ is the sum of squares of the wave functions, the corresponding cut-off radius for the charge density is $2G_{\text{cut}}$. The cut-off radius of the local potential V_{loc} can be regarded the same as that of ρ, because V_{loc} is a functional of ρ. Thus, the cut-off radius of $V_{\text{loc}}\tilde{\Psi}_{n\mathbf{k}}$ is $3G_{\text{cut}}$. As illustrated in Fig. 1, it is sufficient to take the FFT grid with only $2G_{\text{cut}}$ in order to prevent the wave functions from the wrap-around error.

On one hand, the operation of the local potential on trial wave functions are computed with 3-dim FFTs on the standard grid determined by $2G_{\text{cut}}$. On the other hand, we carry out some intensive computations, the time complexities of which are in proportion to the number of plane waves within the cut-off radius G_{cut}, including the assembly of the matrix on the subspace, the orthogonalization of wave functions, and the actions of other parts of the Hamiltonian operator. And the workload of these calculations are not homogeneously distributed on the FFT grid. Therefore, one should consider not only the parallel scaling of FFTs, but also the load balancing issue of the intensive plane wave computations.

3.2. The greedy algorithm

One 3-dim FFT consists of three successive sets of 1-dim FFTs along the x, y and z directions. For each set of 1-dim FFTs, the data layout should guarantee that each processor holds complete columns of data along the FFT direction. Therefore, there are three data layouts of the 1-dim FFTs along the x, y and z directions. We

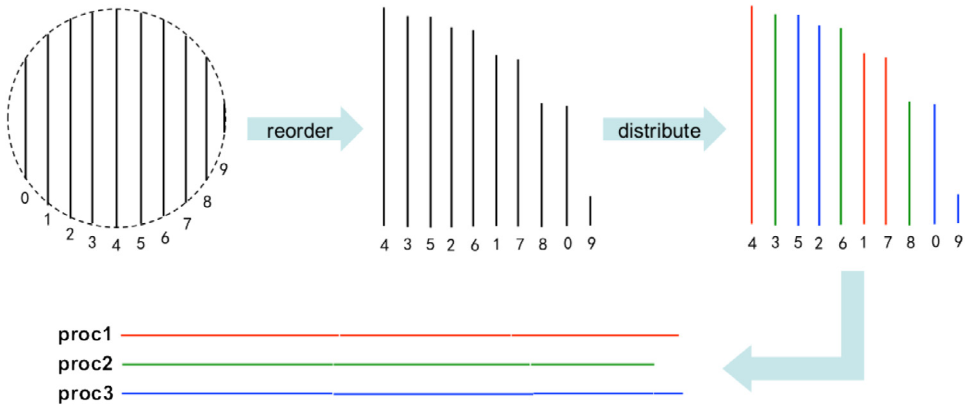


Fig. 2. A two dimensional illustration of the greedy algorithm for building the reciprocal space layout.

call them the reciprocal space, intermediate and real space layouts, respectively.

The greedy algorithm is used to build the reciprocal space layout for the sake of load balancing. In the reciprocal space layout, each processor holds complete columns along the x direction. The workload of each complete column is estimated by the number of plane waves within cut-off radius G_{cut}. As illustrated by Fig. 2, these columns are sorted in the descending order of workload and distributed among processors in a round robin fashion. In this way, the reciprocal space layout is established and each processor holds a set of complete columns with approximately equal workload. For the intermediate and real space layouts, the individual columns could be directly distributed in a cyclic way since the workload of each column is equal.

3.3. *The communication pattern and overhead*

When finishing the set of 1-dim FFTs along one direction, we transpose the data from the current layout to the next one for the successive set of 1-dim FFTs. The first transpose is between the reciprocal space layout and intermediate layout, while the second one is between the intermediate layout and real space layout. With the reciprocal space layout established by the greedy algorithm, the first transpose typically requires the all-to-all communication. The second transpose may require no communications if each processor holds complete planes (perpendicular to the x direction), or limited local communications if each processor has a section of a plane.

In general, the overhead of the all-to-all data communication mainly consists of two parts: the data transmission and the network latency. The transmission cost is proportional to the total size of the data packets, and inversely proportional to the internode bandwidth denoted by β. The latency cost is proportional to the number of data transmissions initiated. We denote the latency of one data transmission by α. It worth noting that α and β are defined for the situation that a node is sending a data packet to some other node and simultaneously receiving a packet from another node.

Without loss of generality, we assume that the all-to-all communications is implemented by the pairwise data exchanges. Alternative implementations can be found in Ref. [16]. Thus the all-to-all communication of p processors can be achieved in $p-1$ phases. In each phase, each processor simultaneously sends a data packet to one processor and receives a packet from another (usually different) processor. Though the sizes of data packets are not uniform (in an Alltoallv operation), the average size of one packet can be estimated as $\mu N_{\mathrm{FFT}}/p^2$, where μ is the size of a single element (typically 16 bytes for a double precision complex data type), and N_{FFT} is the number of degrees of freedom in the FFT grid.

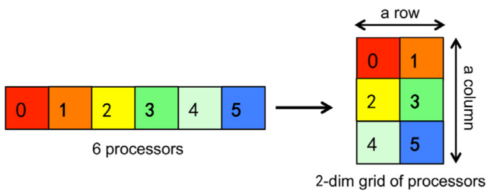


Fig. 3. The illustration of a 3×2 grid of processors.

Hence, we estimate the total cost of one all-to-all communication as

$$t_1 = (p-1)\left(\alpha + \frac{\mu N_{\mathrm{FFT}}}{\beta p^2}\right). \tag{4}$$

For a fixed N_{FFT}, the latency overhead grows linearly with respect to the number of processors, which would probably result in a limited parallel scaling.

4. The new partitioning algorithm and its implementation

In this section, we present a new partitioning algorithm of the 3-dim FFT grid to avoid global all-to-all communications required by the data transposes, so that the latency cost is alleviated at a cost of small loss of load balancing.

4.1. *The partitioning algorithm and its communication overhead*

We assume that the number of processors p can be factorized by $m \times n$, where the difference between m and n, i.e. $|m-n|$, should be as small as possible. Then the p processors are grouped into m rows by n columns (a 3×2 case is illustrated by Fig. 3). Take the reciprocal space layout for example. We distribute the complete columns of data along the x direction following two rules: first, the data columns with the same y-index are distributed within the same column group of processors; second, the data columns with the same z-index are distributed within the same row group of processors. The intermediate and the real space data layout can be established in a similar way. The only restrictions are that the intermediate layout shares the same data distribution with the reciprocal space layout along the z direction and with the real space layout along the x direction. In another word, each xy plane in the intermediate layout is distributed among the same row of processors as the xy plane in the reciprocal space layout with the same z index, and each yz plane in the intermediate layout is distributed among the same column of processors as the yz plane in the real space layout with the same x index.

A direct implementation of the algorithm is to distribute the data columns in a cyclic fashion. As an illustration, we show how

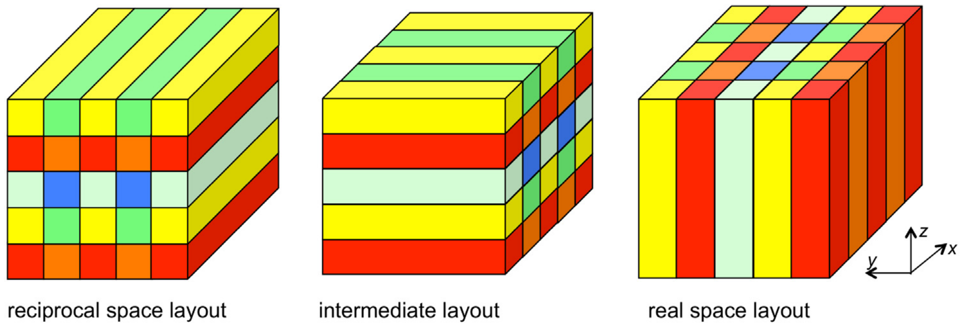


Fig. 4. The resulting three layouts by using the new algorithm to partition the 3-dim FFT grid of $5 \times 5 \times 5$ among 6 processors.

to use it to distribute a 3-dim FFT grid of size $5 \times 5 \times 5$ among 6 processors. The 6 processors are grouped into 3 rows by 2 columns, as shown by Fig. 3. The reciprocal space, intermediate and the real space layouts established by our method are shown, from left to right respectively, by Fig. 4.

In general, the first data transpose (between reciprocal space and intermediate layouts) requires m local all-to-all communications which can be carried out *independently* within row groups of n processors. Similarly, the second transpose (between intermediate and real space layouts) requires n local all-to-all communications which can be carried out *independently* within column groups of m processors. As shown by Fig. 4, the first data transpose requires local all-to-all communications within row groups of two processors, and the second data transpose requires local all-to-all communications within column groups of three processors. When m and n are roughly equal to $\sqrt{p}$, the communication overhead can be estimated as

$$t_2 = (\sqrt{p} - 1)\left(\alpha + \frac{\mu N_{\mathrm{FFT}}}{\beta p \sqrt{p}}\right). \quad (5)$$

Compared with the estimated cost (4) of the global all-to-all communication, the growth rate of the latency cost with respect to the number of processors is decreased from p to $\sqrt{p}$.

It should be clarified that both (4) and (5) are used to qualitatively illustrate how the communication overhead is decreased rather than to give an quantitative interpretation of actual running time. Compared with a global all-to-all communication, the new local all-to-all communications make fewer ($p(\sqrt{p}-1)$ v.s. $p(p-1)$) and larger ($\mu N_{\mathrm{FFT}}/(p\sqrt{p})$ v.s. $\mu N_{\mathrm{FFT}}/p^2$) messages, which alleviates the competition as well as latency issues in the network. So the proposed partitioning algorithm offers the prospect of scaling the plane wave first-principles calculations up to more nodes.

4.2. The load balancing

The local all-to-all communications can be kept if the aforementioned partitioning restrictions are satisfied, i.e. the intermediate layout shares the same data distribution with the reciprocal space layout along the z direction, and the same data distribution with the real space layout along the x direction. So we are allowed to improve the reciprocal space layout considering the load balancing issue.

Here, we just present one tuning approach: firstly, the workload of each xz-plane and xy-plane are estimated by the number of plane waves in the sphere of radius G_{cut}. Secondly, the xz-planes are sorted in the descending order with respect to the workload, and then the reordered xz-planes are distributed to the column groups of processors in a round robin fashion. Finally, the xy-planes are sorted in the descending order with respect to the workload, and then the reordered xy-planes are distributed to the row groups of processors in a round robin fashion.

4.3. The compact 3-dim FFT

As we have discussed in Section 3.1, the standard 3-dim FFT grid is determined by the cut-off radius $2G_{\mathrm{cut}}$, while the wave functions are represented by plane waves within the sphere of radius G_{cut} (Fig. 5(a)). Thus, one can pick up the complete x-data columns that intersect with the sphere to perform 1-dim FFTs along the x direction, because all other x-data columns contain only vanishing values. All the selected columns constitute a cylinder, as shown in Fig. 5(b). After the x direction FFTs, only this cylinder contains non-zero data. Then one can select the complete y-data columns that intersect with the cylinder to perform the y direction 1-dim FFTs, and the resulting non-zero-data region is a cuboid, as shown in Fig. 5(c). The last set of 1-dim FFTs along the z direction is performed on the whole cube shown as in Fig. 5(d). Such a compact 3-dim FFT can also reduce the amount of passing messages and calculations compared to the standard 3-dim FFT implementation that performs 1-dim FFTs on all x and y data columns in the cube.

If only the Γ-point is used for the **k**-point sampling, we can implement a real mode where the reciprocal space and intermediate layouts can be cut by half since we take into account that $\tilde{\Psi}_n(\mathbf{G}) = \tilde{\Psi}_n^*(-\mathbf{G})$.

5. Numerical results

We implement the parallel compact 3-dim FFT for the trial wave functions in the in-house plane wave code package CESSP developed on the infrastructure JASMIN [17]. Our implementation is a pure MPI version including the greedy algorithm, the new algorithm with and without load balancing. With these partitioning algorithms, the parallel scaling of solving Kohn–Sham equation (1) with the PAW approach is tested on a domestic parallel machine. Each node of the machine consists of 2 Intel Xeon E5540 CPUs (8 cores) and the nodes are connected by the infiniband with double data rate (DDR).

The testing systems are face-centered cubic (FCC) supercells consisting of 108 and 500 aluminum (Al) atoms, and is sampled with only the Γ-point. The self-consistent field iteration runs 7 cycles, and in each cycle the RMM-DIIS algorithm [15] is employed to solve the lowest 217 and 1001 eigenstates for the two systems, without the parallelization over bands. In this process, 8246 and 38 038 FFTs of the trial wave functions are executed one by one for the 108 atoms and 500 atoms systems, respectively. The sizes of the 3-dim FFT grid are $48 \times 48 \times 48$ and $80 \times 80 \times 80$, and the numbers of plane waves in the cut-off radius G_{cut} are 7602 and 35 160 for the 108 atoms and 500 atoms systems, respectively.

In all tests, we launch 8 pure MPI processes per node and count the number of data transposes, the total wall time as well as communication time of the trial wave functions' FFTs. The results are summarized in Tables 1 and 2. In the greedy algorithm, no data transposes are required between the intermediate and real space layouts since the intermediate layout holds the complete yz-planes

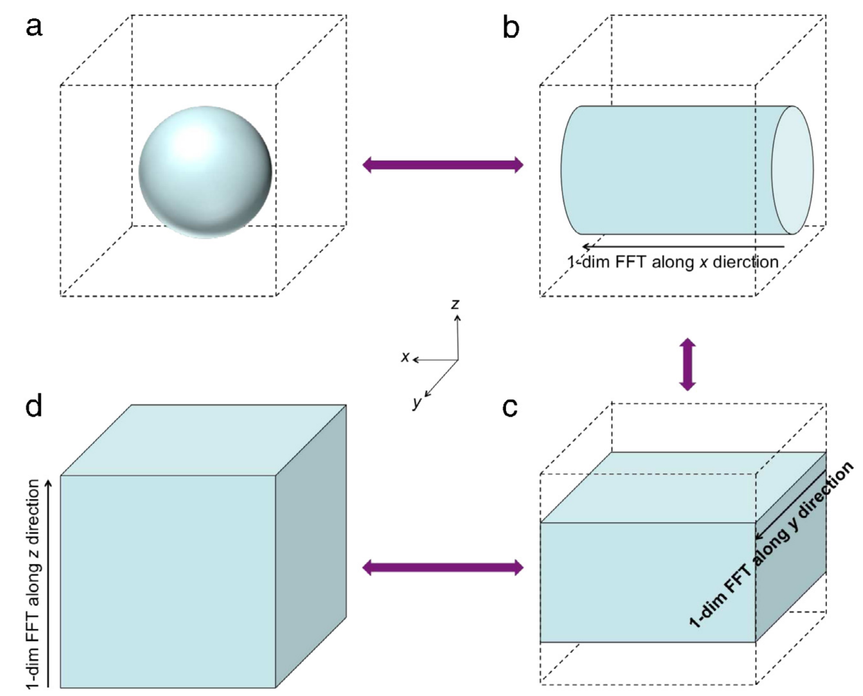

a
b
1-dim FFT along x dierction
z
x
y
d
c
1-dim FFT along z direction
1-dim FFT along y direction

Fig. 5. The illustration of a compact 3-dim FFT. (a): The aqua region presents the sphere of cut-off radius G_{cut}. (b)–(d): The aqua regions present union of all data columns selected to perform the x, y and z 1-dim FFTs, respectively.

Table 1
Comparison on the parallel scaling of three partitioning algorithms for the 108 aluminum atoms system. From left to right: The greedy algorithm, the new algorithm (without load balancing) and the new algorithm with load balancing.

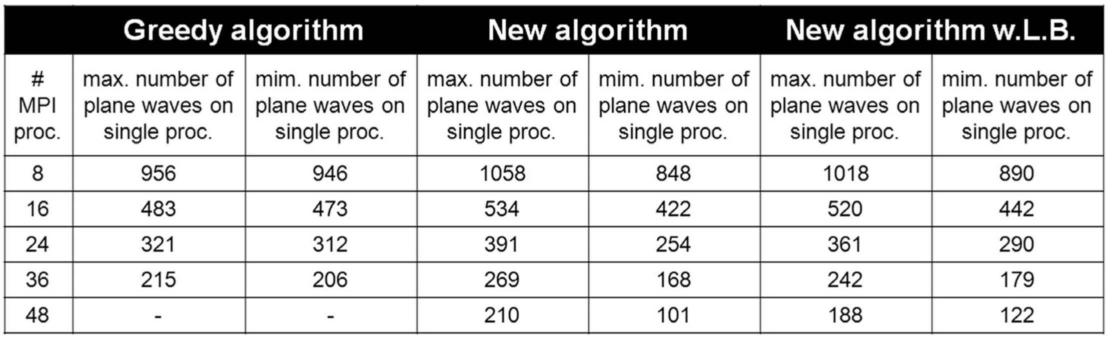
	Greedy algorithm				New algorithm				New algorithm w.L.B.			
# MPI proc.	# data trans-poses	comm. time of trial wfs' FFTs (s)	total wall time (s)	parallel scaling efficiency	# data trans-poses	comm. time of trial wfs' FFTs (s)	total wall time (s)	parallel scaling efficiency	# data trans-poses	comm. time of trial wfs' FFTs (s)	total wall time (s)	parallel scaling efficiency
8	8246	0.51	14.34	1.00	16492	0.92	15.36	1.00	16492	0.99	15.37	1.00
16		0.45	8.26	0.87		0.92	9.11	0.84		0.91	8.79	0.87
24		1.44	8.02	0.60		0.77	7.38	0.69		0.78	7.18	0.71
36		8.14	16.28	0.20		0.55	6.46	0.53		0.64	6.15	0.56
48		-	-	-		0.64	6.62	0.39		0.63	6.33	0.40

Table 2
Comparison on the parallel scaling of three partitioning algorithms for the 500 aluminum atoms system. From left to right: The greedy algorithm, the new algorithm (without load balancing) and the new algorithm with load balancing.

	Greedy algorithm				New algorithm				New algorithm w.L.B.			
# MPI proc.	# data trans-poses	comm. time of trial wfs' FFTs (s)	total wall time (s)	parallel scaling efficiency	# data trans-poses	comm. time of trial wfs' FFTs (s)	total wall time (s)	parallel scaling efficiency	# data trans-poses	comm. time of trial wfs' FFTs (s)	total wall time (s)	parallel scaling efficiency
16	38038	7.61	522.27	1.00	76076	15.59	551.77	1.00	76076	16.61	551.07	1.00
24		9.28	373.82	0.93		13.40	388.44	0.95		14.42	370.30	0.99
36		35.11	291.53	0.80		10.36	275.51	0.89		9.78	256.14	0.96
64		117.32	303.31	0.43		8.62	183.21	0.75		7.79	176.13	0.78
96		-	-	-		10.07	171.73	0.54		9.73	162.58	0.56
128		-	-	-		13.63	169.09	0.41		12.88	157.61	0.44

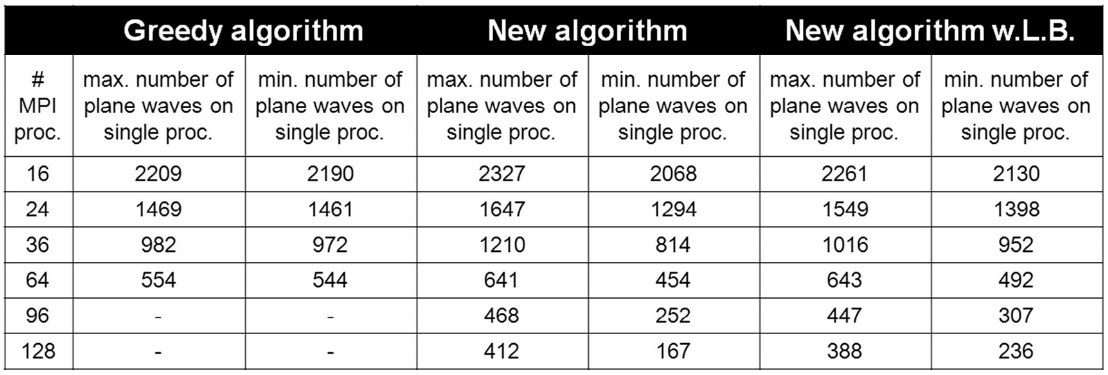
on each processor. So the greedy algorithm needs half number of data transposes as the new algorithm. Nevertheless, as shown by Tables 1 and 2, with an increasing number of processors, the greedy algorithm leads to a rapid growth in the communication cost, which finally takes a substantial part of the total computational cost, while the new algorithm (with or without load balancing) can effectively suppress the growth in the communication cost. When the number of processors is less than 24, the greedy algorithm is preferable, while when the number of processors is more than 24, the new algorithm could provide better overall performance.

Table 3
Comparison on the load balancing of three partitioning algorithms for the 108 aluminum atoms system. From left to right: The greedy algorithm, the new algorithm (without load balancing) and the new algorithm with load balancing.



	Greedy algorithm		New algorithm		New algorithm w.L.B.	
# MPI proc.	max. number of plane waves on single proc.	mim. number of plane waves on single proc.	max. number of plane waves on single proc.	mim. number of plane waves on single proc.	max. number of plane waves on single proc.	mim. number of plane waves on single proc.
8	956	946	1058	848	1018	890
16	483	473	534	422	520	442
24	321	312	391	254	361	290
36	215	206	269	168	242	179
48	-	-	210	101	188	122

Table 4
Comparison on the load balancing of three partitioning algorithms for the 500 aluminum atoms system. From left to right: The greedy algorithm, the new algorithm (without load balancing) and the new algorithm with load balancing.



	Greedy algorithm		New algorithm		New algorithm w.L.B.	
# MPI proc.	max. number of plane waves on single proc.	min. number of plane waves on single proc.	max. number of plane waves on single proc.	min. number of plane waves on single proc.	max. number of plane waves on single proc.	min. number of plane waves on single proc.
16	2209	2190	2327	2068	2261	2130
24	1469	1461	1647	1294	1549	1398
36	982	972	1210	814	1016	952
64	554	544	641	454	643	492
96	-	-	468	252	447	307
128	-	-	412	167	388	236

In Tables 3 and 4, we represent the load balancing in the simulations by comparing the maximum and minimum numbers of plane waves distributed on a single processor. It is obvious that the load balancing of the greedy algorithm is almost perfect. The load balancing strategy for the new partitioning algorithm (Section 4.2) effectively reduces the gap between the maximum and minimum numbers of plane waves on the processors. Although not as perfect as the greedy algorithm, it is acceptable for practical simulations. The benefit from the improved load balancing is not significant for the 108 atoms system. For the 500 atoms system, the load balancing saves up to 19 s in the total wall time. What we present in Tables 1–4 is an example of the trade-off between the load balancing and the communication cost: the greedy algorithm has the best load balancing but could lead to very limited parallel scaling; the new algorithm achieve much better scaling at a moderate loss of load balancing.

It should be noted that our code provide three levels of parallelization: the FFT grids, bands and k-points. In practical simulations, all three levels will be used if possible. In the presented numerical examples, in contrast, we only switched on parallelization over FFT grids, because the focus of the manuscript is to develop a new parallel FFT strategy, and we intend to eliminate the influence on the performance from other levels of parallelization.

6. Conclusion

We propose a new partitioning algorithm for the 3-dim FFT grid used in the plane wave first-principles calculations. Compared with the greedy algorithm biased toward the load balancing of the plane wave computations, our approach primarily suppresses the growth in communication overhead with respect to an increasing number of processors by realizing local all-to-all communications during data transposes. Then we adjust the data distribution to improve the load balancing with the communication pattern preserved. It has been shown by numerical results that a much lower communication overhead on a relatively large number of processors is achieved at a moderate loss of load balancing. Using the new algorithm, we could scale the plane wave codes up to more nodes than the greedy algorithm. If better performance were wanted, we would combine our approach with other techniques such as the hybrid OpenMP/MPI implementation or simultaneously performing a large number of FFTs.

Acknowledgments

The authors would like to thank Hong Guo and Zhang Yang for useful discussions. And the first author is especially thankful to Xiaowen Xu for his encouragement. This work was partially supported by the National Science Foundation of China under Grants 91430218 and 61300012, and the National High Technology Research and Development Program of China under Grant 2015AA01A304. H.W. is supported by the National Science Foundation of China under Grants 11501039 and 91530322.

References

[1] W. Kohn, L.J. Sham, Phys. Rev. 140 (1965) A1133.
[2] G. Kresse, J. Furthmüller, Comput. Mater. Sci. 6 (1996) 15.
[3] G. Kresse, J. Furthmüller, Phys. Rev. B 54 (1996) 11169.
[4] M.C. Payne, M.P. Teter, D.C. Allan, Rev. Modern Phys. 64 (1992) 1045.

[5] B. Liu, in: E. Moler, I. Shavitt (Eds.), Numerical Algorithms in Chemistry: Algebraic Methods, Lawrence Berkley Lab. Univ. of California, 1978, p. 49.
[6] A.V. Knyazev, SIAM J. Sci. Comput. 23 (2001) 517.
[7] Y. Zhou, Y. Saad, M.L. Tiago, J.R. Chelikowsky, J. Comput. Phys. 219 (2001) 172.
[8] S. Goedecker, M. Boulet, T. Deutsch, Comput. Phys. Comm. 154 (2003) 105.
[9] A. Canning, J. Shalf, N.J. Wright, S. Anderson, M. Gajbe, The 9th International Conference on Scientific Computing, 2012.
[10] P.D. Haynes, M. Côté, Comput. Phys. Comm. 130 (2000) 130.
[11] G.C. Danielson, C. Lanczos, J. Franklin Inst. 233 (1942) 365.
[12] D.R. Hamann, M. Schlüter, C. Chiang, Phys. Rev. Lett. 43 (1979) 1494.
[13] D. Vanderbilt, Phys. Rev. B 41 (1990) 7892.
[14] P.E. Blöchl, Phys. Rev, B 50 (1994) 17953.
[15] G. Kresse, J. Joubert, Phys. Rev. B 59 (1999).
[16] L. Rao, Y. Zhang, Y. Li, Comput. Sci. 37 (2010) 186.
[17] Z. Mo, A. Zhang, X. Cao, Q. Liu, X. Xu, H. An, W. Pei, S. Zhu, Front. Comput. Sci. China 4 (2010) 480.