TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 07/12 pp309–321
DOI: 10.26599/TST.2019.9010075
Volume 26, Number 3, June 2021

Parallel Optimization of the Crystal-KMC on Tianhe-2

Jianjiang Li, Baixue Ji, Yun Yang, Peng Wei*, and Jie Wu

Abstract: The Kinetic Monte Carlo (KMC) is one of the commonly used methods for simulating radiation damage of materials. Our team develops a parallel KMC software named Crystal-KMC, which supports the Embedded Atom Method (EAM) potential energy and utilizes the Message Passing Interface (MPI) technology to simulate the vacancy transition of the Copper (Cu) element under neutron radiation. To make better use of the computing power of modern supercomputers, we develop the parallel efficiency optimization model for the Crystal-KMC on Tianhe-2, to achieve a larger simulation of the damage process of materials under irradiation environment. Firstly, we analyze the performance bottleneck of the Crystal-KMC software and use the MIC offload statement to implement the operation of key modules of the software on the MIC coprocessor. We use OpenMP to develop parallel optimization for the Crystal-KMC, combined with existing MPI inter-process communication optimization, finally achieving hybrid parallel optimization. The experimental results show that in the single-node CPU and MIC collaborative parallel mode, the speedup of the calculation hotspot reaches 30.1, and the speedup of the overall software reaches 7.43.

Key words: Kinetic Monte Carlo (KMC); Tianhe-2; parallel optimization; OpenMP

1 Introduction

The physical properties of metallic materials are relatively stable. In the normal state, the atoms continue to perform thermal motion. However, most atoms oscillate near the minimum in the potential energy surface region, and atomic transition events are less likely to occur. Only extreme conditions, such as neutron irradiation, can cause certain atoms to obtain enough energy to exceed the potential energy barrier and complete transition. The accumulation of these transitions is the main reason for the change in material physical properties. At present, simulation tools are widely used to predict the effects of neutron irradiation on pressure vessels and containment shells by utilizing the powerful computing power of supercomputers[1].

The Kinetic Monte Carlo (KMC) is the most common classical method for simulating radiation damage of materials. The KMC method[2] obtains the transition of the system by calculating the probability of events using random numbers. Therefore, in the KMC simulation, the focus is not on the atom but on the system. Thus, the simulation time is an exponential factor of the system configuration transition. The KMC method can describe the dynamic characteristics of the system. The time scale and spatial scale of the simulation are relatively large, and the long-term simulation can be completed. The interaction force between atoms of each substance has different characteristics, so a potential energy function can be introduced to assist computers to carry out material evolution more accurately. The simulated materials in this paper are mainly composed of Body-Centered Cubic (BCC) structure Reactor Pressure Vessel (RPV) steel[2] (Fe-based alloy containing impurities,

• Jianjiang Li, Baixue Ji, and Yun Yang are with the Department of Computer Science and Technology, University of Science and Technology Beijing, Beijing 100083, China. E-mail: jianjiangli@163.com; baymaxuer@163.com; yangyun_1110@163.com.
• Peng Wei is with Industrial and Commercial Bank of China Shandong Branch, Jinan 250001, China. E-mail: weissnh@163.com.
• Jie Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA. E-mail: jiewu@temple.edu.
∗ To whom correspondence should be addressed.
Manuscript received: 2019-10-08; accepted: 2019-12-05

such as Cu, Ni, and Mn, etc.). For this material, the Embedded Atom Method (EAM) potential[3] is used to describe the evolution characteristics of the system accurately.

Hoffmann et al.[4] developed a dynamic Monte Carlo serial open source software called kmoss. They studied microfacies in complex reaction networks by simulating the basic processes that occurred at the active sites of static lattices. Domain et al.[5] simulated the radiation damage of RPV steel by LAKIMOKA, which used Object Kinetic Monte Carlo (OKMC) method. Borodin et al.[6] used the CASINO-LKMC program to simulate the formation process of helium-vacancy clusters in iron with a BCC structure.

KMC is essentially a serial algorithm; that is, the system should be in a specific state at a certain time and changes over time to transitions between states. However, serial computing is slow and limits the time and space resources available for simulation. The emergence of large parallel computers, distributed shared memory processors, and PC-Cluster, etc., has made more computing resources available. They support long-timescale and large-scale KMC simulations. At present, the research on KMC parallel algorithm has become one of the most important research directions. Owing to this effect, many scholars have studied and proposed some KMC parallel simulation algorithms or frameworks.

Shim and Amar[7] proposed the synchronous Sulrlattice (SL) algorithm to avoid conflicts between processors. On this basis, Martínez et al.[8] proposed a rejected KMC parallel algorithm based on domain decomposition, which introduced empty events into each subdomain to ensure that the total transition rate of each subdomain was the same.

An open source KMC parallel framework SPPARKS was developed by the Sandia National Laboratories, USA. In addition to rejected KMC, it also provided non-rejected KMC and Multicanonical Monte Carlo (MMC) algorithms[9]. SPPARKS is a widely used parallel KMC software framework that supports multiple potential functions. However, it does not support the EAM potential[10], so it cannot be used to simulate the radiation damage of nuclear cladding materials.

MMonCa is an OKMC simulation software written by Martin-Bragado et al.[11] It provides thread parallelism for two-axes region segmentation without communication between parallel regions. Leetmaa and Skorodumova[12] implemented a lattice dynamics simulation parallel framework called KMCLib, which is easy to customize and integrate with other software.

Many research groups have independently designed new physical models to simulate a particular type of system; thus, it is difficult to develop a parallel software that satisfies all KMC applications. Both SPPARKS and KMCLib show better parallelism and scalability, but in addition to the KMC applications they have implemented, users need to implement the program interfaces according to their physical models. Also, because of the multi-model compatibility needs, platform software is more complicated in terms of logic and implementation than dedicated software.

To meet the needs of simulating the specific environment of irradiation damage, our research group has independently developed a parallel KMC simulation software called Crystal-KMC[13]. The parallel software is based on lattice dynamics; it supports EAM potential functions and takes full advantage of the characteristics of data sharing in shared memory. To meet the needs of large-scale applications, we optimize the Crystal-KMC at different levels and scales based on the architecture of the Tianhe-2 supercomputer.

The rest of the paper is organized in the following manner. Section 2 gives a brief introduction to Crystal-KMC. Section 3 elaborates on the performance optimization of Crystal-KMC on Tianhe-2 (including performance bottleneck analysis, kernel optimization, OpenMP optimization, and overall hybrid optimization). Section 4 is the analysis of the experiment and results, and Section 5 summarizes the study.

2 Introduction to Crystal-KMC

The KMC simulation is a method used to simulate the long-term evolution of systems by constructing a stochastic process. It is also used in the Crystal-KMC software. This method has been implemented by various algorithms, such as the random selection method algorithm, the Gillespie Stochastic Simulation Algorithm (SSA), and the Variable Step-Size Algorithm (VSSA), etc.[14] Its main uses are for the simulation of crystal surface diffusion[15], metal surface gas adsorption[16], thin film growth[17], grain growth[18], and material irradiation damage[4]. The basic idea of the method is: firstly, to determine the initial state of the system which includes the number of events M and the related velocity constant K_{ij}, which changes with each event. Then the number generated by the random number generator determines the time t needed to change from the initial state to the next state, and the events (paths) selected by the process. After the corresponding changes occur, the system will transit from the initial state to the next state. This method can simulate a longer time scale, which is usually in seconds or longer.

The Crystal-KMC, a KMC parallel software developed by our group, is mainly used to simulate the aging of nuclear cladding materials under fast neutron irradiation. The software supports the EAM potential function, customizes radiation damage of nuclear materials, and innovatively proposes an optimization method to reduce the communication overhead[13]. Figure 1 shows the basic process of Crystal-KMC performing transition simulation within a certain time threshold. Note that in Fig. 1, "Time" means the time threshold, r_1 and r_2 are two random numbers, ranging from 0 to 1, which are used to calculate the transition path and the simulation time required for the transition,

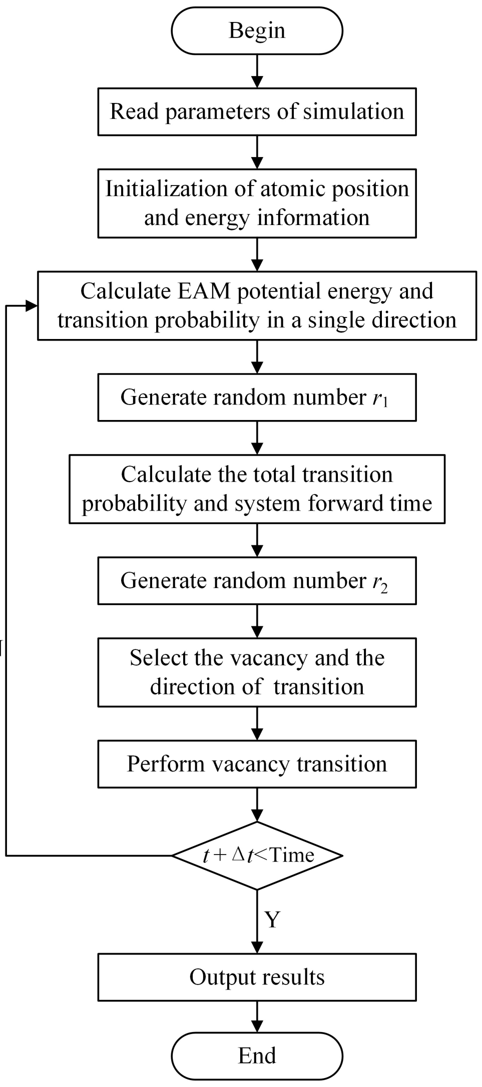

Begin
Read parameters of simulation
Initialization of atomic position and energy information
Calculate EAM potential energy and transition probability in a single direction
Generate random number r_1
Calculate the total transition probability and system forward time
Generate random number r_2
Select the vacancy and the direction of transition
Perform vacancy transition
$t+\Delta t$<Time
N
Y
Output results
End

Fig. 1 Basic flow of Crystal-KMC performing transition simulation within a certain time threshold.

respectively. The initial value of t is 0, and the value of Δt is 1.25×10^{-8} s.

The Crystal-KMC software is divided into ten modules. However, the energy module, the communication module, and the calculation transition module (simulation) are repeatedly called and executed. From Fig. 2, we can see that the most time-consuming module is the computation transition module (simulation). With the expansion of scales, the time-consuming property of the module of energy, simulation, and communication increases gradually. If the scale is more significant, then the run time increases. It is important to note that the communication part has been significantly optimized in the Crystal-KMC such that it will no longer be optimized in this paper. Therefore, we focus on the energy module and the simulation module. For the energy module, we use task parallel optimization. Whereas for the simulation module, we use task-parallel optimization and data-parallel optimization.

3 Crystal-KMC Optimization on Tianhe-2

Since the development of high-performance computing technology, it has been widely applied in geophysics, atmospheric meteorology, industry, materials, and so on. High-performance computing has become an important research tool in science and technology. This section focuses on the optimization of the Crystal-KMC on Tianhe-2 supercomputer.

3.1 Performance bottleneck analysis

Firstly, parallel optimization for the Crystal-KMC requires finding out the performance bottleneck of the Crystal-KMC software. The Crystal-KMC contains many modules, and the amount of code is huge. Therefore, we take the module as the object and run the Crystal-KMC to simulate the diffusion of vacancies in iron on Tianhe-2 with single process and single thread for different test scales, which are shown in Table 1. The simulated space size represents the length, width, and

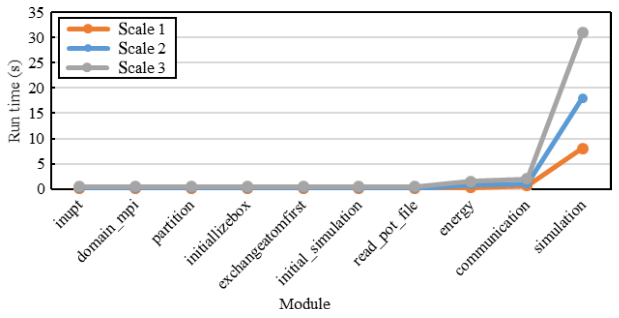


Fig. 2 Run time of modules in Crystal-KMC.

Table 1 Different test scales.

Scale No.	Simulated space size	Vacancy count	Loop count
1	$40 \times 40 \times 40$	400	20
2	$40 \times 40 \times 40$	800	20
3	$80 \times 80 \times 80$	800	20

height of the simulated space. The running result is shown in Fig. 2.

From Fig. 2, three time-consuming modules in the Crystal-KMC are the transition computing module (simulation), the communication module, and the energy update module. The time-consuming property of the communication module is mainly caused by the overhead of Message Passing Interface (MPI) communication. In the previous work, the MPI communication of Crystal-KMC software has been optimized[13]. Therefore, we focus on the optimization of computing transition module and the energy update module.

The energy update module is performed after each execution of the KMC simulation in the Crystal-KMC software. Thus, the KMC and its corresponding global energy updates execution time are proportionally related to the total computation time by the predefined time threshold. Hence, the higher the time threshold, the larger the proportion of the execution time of KMC to the total computing time, and the smaller the proportion of the computation time corresponding to energy update to the total computing time.

For the small-scale test, the transition computing module (simulation) in Crystal-KMC is analyzed by using Intel VTune$^{\text{TM}}$ Amplifier XE[19]. The results show that the main time-consuming functions in the module are energy() and calcul_proba(). And calcul_de() accounts for more than 90% of the total run time, as shown in Table 2.

Table 2 Run time of functions in simulation module.

Function	Module	CPU time (s)	Overhead time (s)	Spin time (s)
calcul_proba()	Simulation	783.23	0	0
calcul_de()	Simulation	167.45	0	0
energy()	Simulation	152.49	0	0
eval_embed()	Simulation	31.56	0	0
read_pot_file()	Simulation	22.18	0	0
eatom()	Simulation	13.45	0	0
choix_ vaca-ncy_jump()	Simulation	7.56	0	0
onr_ deplac-ement()	Simulation	4.32	0	0
compute()	Simulation	2.78	0	0
inc_px()	Simulation	0.36	0	0

The main function of energy() is to calculate the global energy update after the transition (the energy update module of Crystal-KMC). The performance bottleneck is that the vacancy energy needs to be calculated independently in the triple nested for loop. The calcul_proba() calculates the probability of a transition event, and the calcul_de() calculates the amount of change in system energy before and after the transition event. These parts are called many times and in the loop at the same time, which can be optimized in parallel. Table 3 describes the time-consuming functions calcul_proba() and calcul_de().

In the process of program execution, calcul_proba() is called eight times for each KMC step. calcul_de() contains two functions, calcul_dene_v1() and calcul_dene_v2(), whose execution time is basically the same. In the process of program execution, calcul_de() calls calcul_dene_v1() twice as many times as calcul_dene_v2() calls.

After analyzing the performance of Crystal-KMC, we deal with the core code in the coprocessors on Tianhe-2 using offload mode. Within each process, tasks are

Table 3 Introduction to the time-consuming functions calcul_proba() and calcul_de().

Function	Feature	Parameter	Process of function
calcul_proba()	Calculate the probability of a transition event.	$(i,\ j,\ k)$ position of $(d_i,\ d_j,\ d_k)$ vacancy transition direction.	(1) Calculate the position of vacancy transition (the three dimensions are similar); (2) Get the entity type (atoms/vacancies) of the transition destination; (3) Calculate the amount of change in potential energy before and after the transition–call the calcul_de(); (4) Calculate the probability of transitions.
calcul_de()	Calculate the amount of change in system energy before and after the transition event.	$(i,\ j,\ k)$ position of $(d_i,\ d_j,\ d_k)$ vacancy transition direction.	(1) Calculate the coordinates of the nearest neighbor; (2) Calculate energy based on the potential energy model.

divided into threads of a multi-core coprocessor. Typical transition computing simulation carries out global inter-process communication and global energy update after each vacancy transition. So, in each time step, the data exchange between the main core and the coprocessor will take place. The multiple *for loop* operations between data exchanges can be performed on the Many Intergrated Core (MIC). In the Crystal-KMC computation process, some data remain unchanged during the entire simulation process. This part of data only need to be offloaded once and transmitted to the coprocessor.

3.2 Parallel optimization of the energy update computation

3.2.1 Splitting of the energy update loop

It is necessary to eliminate data dependencies in the loop area to parallelize efficiently. Data dependency refers to the fact that the current iteration value of a variable in a *for loop* depends on the result of the previous iteration. The parallelization of nested loops can be realized by loop transformation methods, such as loop splitting. Here, the variables with data dependence can be separated from the main body of the *for loop* to form another loop (which is serially executed by the loop), and the main loop without data dependence can be executed in parallel.

From the result of Section 3.1, the energy() function calculates the vacancy energy one by one. The function made up of a *for loop* which traverses the interaction force between each atom and the surrounding atoms in the simulation region. The force beyond the truncation radius is ignored. The EAM potential energy of each atom interacting with those atoms within its intercept radius is calculated by the eatom() function. Algorithm 1 shows the simplified EAM potential energy computation pseudocode.

In each cell, there are many atoms, and the energy of all atoms and other information are stored in a three-dimensional array EV[][][], while pos is used to locate and retrieve the data of an atom. As iterations are made in the loop, the variable pos will accumulate accordingly, causing a data-dependent loop. To parallelize the loop, it is necessary to split the code fragment with data dependencies out of the main loop body. The accumulation of pos variables is thus separated into a new loop, and an array (array[]) is created to hold the values of pos after the k-th iteration. Then the main loop which contains the EAM potential energy computation is transformed into a double nested loop, in which the pos stored in array[] is used directly.

As shown in Algorithms 2 and 3, there is no data dependency on the pos in the main loop. Therefore, the main loop in the EAM potential energy computation after loop split optimization can be executed in parallel.

3.2.2 Crystal-KMC acceleration on MIC using offload mode

The Tianhe-2 supercomputer is a heterogeneous platform. Its processors include CPU and MIC cards which can be combined to improve performance. The MIC card can be used as a coprocessor to perform computationally intensive tasks, or as a common node equivalent to CPU. We make full use of MIC cards with outstanding computing power to perform computationally intensive tasks and to improve the performance of programs on a single node.

At present, the offload mode is the most common Tianhe-2 working mode in use. Therefore, we also use offload mode to accelerate the Crystal-KMC

Algorithm 1 EAM potential energy computation

```
1: for cell $c = 0$ to max //max is the size of space box
2:     for atom $i$ in cell $c$
3:         pos=pos+size;
4:         ...
5:         call eatom(); // performing EAM potential energy operations
6:         ...
7:     end for
8: end for
```

Algorithm 2 New loop: Loop split optimization for EAM potential energy computation

```
1: for cell $c = 0$ to max //max is the size of space box
2:     for atom $i$ in cell $c$
3:         pos=pos+size;
4:         array[$k$]=pos;
5:     end for
6: end for
```

Algorithm 3 Main loop: Loop split optimization for EAM potential energy computation

```
1: for cell $c = 0$ to max //max is the size of space box
2:     for atom $i$ in cell $c$
3:         pos = array[$k$];
4:         call eatom(); //performing EAM potential energy operations
5:     end for
6: end for
```

computation on the MIC card. After defining variables and eliminating data dependency, specific statements are needed to specify which processor code is executed and how data are mapped between host memory and accelerator memory. This mode requires executions to be offloaded to the MIC card. Thus, for the energy computation part, it is necessary to offload the energy computation nested loop that traverses all the information onto the MIC card.

Similar to OpenMP, MIC programming[20] also adds compilation instructions, hiding a lot of details, and needs no extra space. The MIC offload compilation instructions in this section are shown in Algorithm 4.

"#pragma offload target" is an MIC offload compiler instruction statement. When the program executes the instruction statement, if there is a corresponding compiler, the code segment related to the instruction statement will be compiled and offloaded to the MIC card for execution. Otherwise, the instruction statement will be ignored. "#pragma end offload" means the end of offload mode. The "nocopy" keyword indicates only open or reserve space on MIC, and no data are transmitted; "in", "out", and "inout" are responsible for data transmission between CPU and MIC. "in" means that when entering the offload area, the array is transferred from CPU memory to MIC memory, only incoming and not outgoing. "out" means that at the end of the offload area, the data are returned from MIC to CPU memory, only outgoing and not incoming. "inout" means that when entering the offload area, the array is transferred from CPU memory to MIC memory, and when leaving, the array is returned. The detailed meaning is shown in Table 4.

By analyzing the code of energy calculation, the data can be divided into three categories: (1) The first type of data only runs on the CPU, which need not be processed; (2) the second type of data is permanent data on the MIC, which remains unchanged throughout the simulation process. For this part of the data, it is offloaded to the MIC card before the iteration begins. For example, for static data in energy computation, such as nghost1, T (temperature), EAM potential energy table, and near-atom index, it is necessary to use "nocopy" to open up space before an offload to MIC on CPU. (3) The third type of data is frequently exchanged between CPU and MIC. These data are usually global variables, which need to be stored in both CPU and MIC to ensure the normal flow of data.

The code for the energy computation section is shown in Algorithm 5.

Lines 1–3 declare that the eatom() function will be executed on MIC. Lines 4–7 indicate there is

Algorithm 4 MIC offload compilation instructions

```
1: #pragma offload target (mic:0)\
2:     nocopy(p_s:length(mic_thread*nv*3)alloc_if(.true.)free_if(.false.))\
3:     in(nghost1:length()alloc_if(.true.true)free_if(.false.))
4:     ...
5: #pragma end offload
```

Table 4 Typical MIC offload statement.

Offload statement	Function
#pragma offload target	Specify the code area that needs to be mapped and executed by mapping the host buffer to the MIC device area. After execution, it is transferred back to the host area by the MIC device area.
#pragma offload_attribute (push, target (mic))	Declare functions that are executed on the target device. Used in pairs with #pragma offload_attribute(pop).
#pragma offload target in (out/inout)	Used to map host-side data to the MIC device. The target area can be executed in the device data environment.
#pragma offload target nocopy	Only open or reserve space on MIC.

Algorithm 5 MIC parallel of energy computation in Crystal-KMC (using offload mode)

NUM_THREAD: Number of threads participating in parallel computing
ene: energy of each atom
enetot: total energy

```
1: #pragma offload_attribute (push, target(mic))
2: void simulation::eatom(...);
3: #pragma offload_attribute(pop)
4: #pragma offload target (mic: 0)\
5: nocopy (p_c: length(mic_thread*nv*3) alloc_if(.true.) free_if(.false.))...\
6:   in (nghost1: length () alloc_if(.true.true)free_if(.false.))...\
7: out (enetot: length () alloc_if(.true.true)free_if(.false.))
8: #pragma omp parallel for private ($it, i, j, k$, cc, $i_0$, $j_0$, $k_0$, e_v, e_r, e_s, ene) reduction(+:enetot)
9: {
10:     for($i$ = 0; $i$ <natom; $i$++){
11:         Calculate ene;
12:         enetot+=ene;
13:     }
14: }
15: #pragma end offload
```

space on the MIC, CPU data are mapped to the MIC side, and the target area can be executed in the device data environment. Line 8 indicates that the OpenMP statement declares a private variable, the energy computation, and the reduction of total energy. Line 15 indicates the end of the offload on the MIC card.

3.3 Performance optimization of computing transition module

During the Crystal-KMC software execution, the calcul_proba function is called eight times for each KMC step. This is because, for materials with the BCC crystal structure, the transition probability for each vacancy is the sum of the probability of transitions to its eight adjacent grid points. Every KMC process needs to calculate the transition probabilities in eight directions for all vacancies in the current computational area, and the computation of transition probabilities in these eight directions is independent of each other and does not share data dependency, so this can be executed in parallel.

Algorithm 6 shows multi-thread parallelism for computing the probability part of the Crystal-KMC software (using OpenMP).

After calculating the transition probability of vacancies, it is necessary to calculate the transition of each vacancy, and the calculation transition of each vacancy is independent. Therefore, these vacancies can be allocated to several threads for parallel execution. At the end of the parallel region, the calculation results of each thread need to be summed up.

Figure 3 shows the flowchart for parallel computation of the EAM potential and the corresponding forces in Crystal-KMC in offload mode. The initial value of t is 0, and the value of Δt is 1.25×10^{-8} s. Algorithm 7 shows MIC parallel for vacancy transition computing in Crystal-KMC.

3.4 CPU+MIC collaborative parallelism

After optimizing the offload mode, the hotspot computation by the transition module was implemented on the MIC card. Its task flowchart is shown in Fig. 4.

In optimization mode, when executing to the offload area, its operations will be performed on the MIC (right area in Fig. 4). At this point, the left area in Fig. 4 is suspended and the CPU is idle. When the execution time on the MIC is long, the CPU will always be in a waiting state. In order to make full use of the computing power of the Tianhe-2 CPU, we use the master/slave

Algorithm 6 Multi-thread parallel probability computing in Crystal-KMC

NUM_THREAD: Number of threads participating in parallel computing

tmpPlist[*n*]: store transition probability of vacancies in *n* directions

tmpDElist1[*n*]: store energy changes caused by transition of vacancies in *n* directions

```
1: #pragma parallel sections
2: {
3:     #pragma section
4:     {
5:         tmpPlist[1] =calcul_proba(x, y, z, 1, 1, 1, ii_fia, &dene);
           //calculate the probability of direction 1
6:         tmpDElist1[1]= dene; // energy change in direction 1
7:         #pragma omp critical
8:         {
9:             (*Plist).push_back(tmpPlist);
10:            (*DElist).push_back(tmpDElist);
11:        }
12:    }
       ...
13:    #pragma section
14:    {
15:        tmpPlist[8] = calcul_proba(x, y, z, –1, –1, –1, ii_fia, &dene); // calculate the probability of direction 8
16:        tmpDElist[8] = dene; // energy change in direction 8
17:    }
18:    #pragma omp critical
19:    {
20:        (*Plist).push_back(tmpPlist);
21:        (*DElist).push_back(tmpDElist);
22:    }
23: }
```

cooperative parallel model[21].

In the nested loop of the energy computation part, the EAM potential energy computation is performed by dividing the cell, whereas the atomic potential energy is calculated by calling “eatom()” in the loop body. Therefore, from the previous analysis, the outer loop traverses the cell, while the inner loop traverses all atoms in the cell, while that of the inner loop is to traverse all atoms in the cell. Since the data dependence between loops has been eliminated, we use task partitioning to run part of the cell’s computing tasks on CPU. The MIC compiling instructions “signal” and “wait” are used to achieve master/slave cooperative parallelism, as shown in Algorithm 8. Firstly, by adding a “signal” statement,

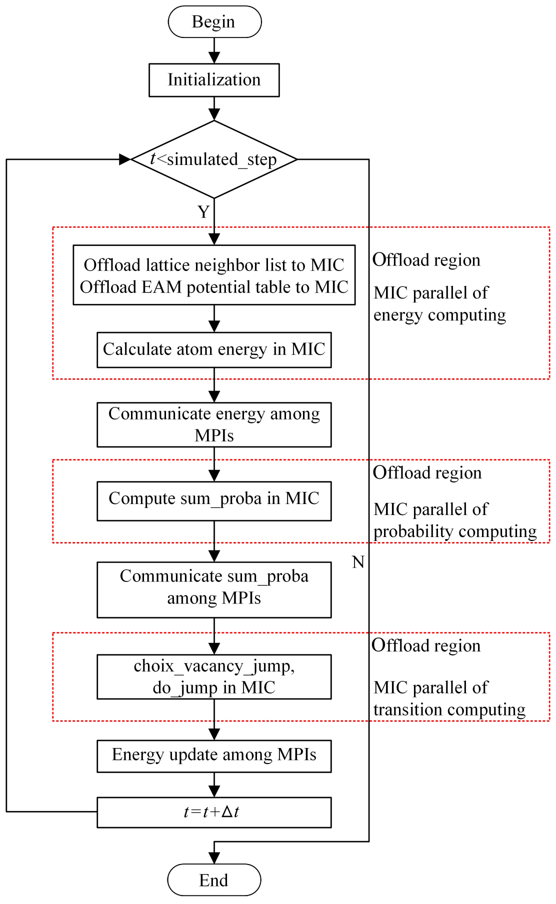

Begin
Initialization
t<simulated_step
Y
Offload lattice neighbor list to MIC
Offload EAM potential table to MIC
Offload region
MIC parallel of energy computing
Calculate atom energy in MIC
Communicate energy among MPIs
Offload region
Compute sum_proba in MIC
MIC parallel of probability computing
N
Communicate sum_proba among MPIs
Offload region
choix_vacancy_jump, do_jump in MIC
MIC parallel of transition computing
Energy update among MPIs
t=t+Δt
End

Fig. 3 Flowchart for the parallel computation of the EAM potential and the corresponding forces in Crystal-KMC (offload mode).

Algorithm 7 MIC parallel for vacancy transition computing in Crystal-KMC (using offload mode)

NUM_THREAD: Number of threads participating in parallel computing
Sum_proba: Transition probability
do_jump(): Vacancy transition

```
1: #pragma offload_attribute (push, target (mic))
2: void simulation::do_jump (...);
3: choix_vacancy_jump();
4: #pragma offload_attribute (pop)
5: #pragma offload target (mic:0)\nocopy()
...
6: out (sum_proba: length ()alloc_if(.true.true)free_if(.false.))
7: #pragma omp parallel private $(x, y, z)$ firstprivate (numth, Plist, DElist, ...)
8: {
9:     #pragma omp for reduction (+:sum_proba) reduction (+:numnv)
10:        for(iv=0; iv<nv; iv++){
11:            choix_vacancy_jump; // do vacancy transition
12:        }
13: }
```

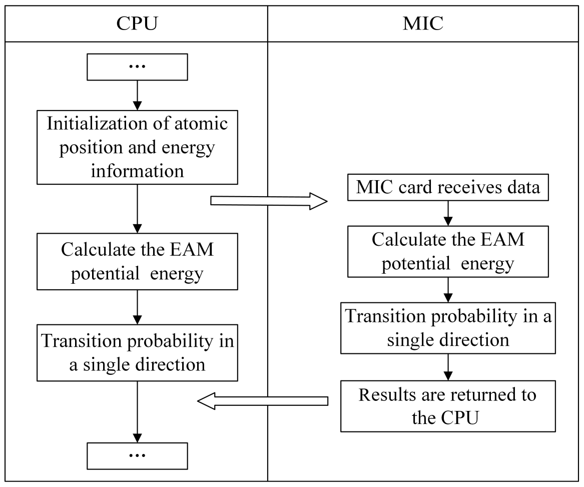

CPU
MIC
...
Initialization of atomic position and energy information
MIC card receives data
Calculate the EAM potential energy
Calculate the EAM potential energy
Transition probability in a single direction
Transition probability in a single direction
Results are returned to the CPU
...

Fig. 4 Flowchart after MIC optimization.

Algorithm 8 CPU+MIC collaborative parallel for optimization mode computing in Crystal-KMC

```
#pragma offload target (mic:0) signal (signal in1)
for($c$ = 0; $i$ <max-$k$; $c$++)
{
    mic_compute(); // code in MIC
}
#pragma end offload
for($c$ =max-$k$; $c$<max; $c$++)
{
    cpu_compute(); // CPU+MIC Collaborative Parallelism (Perform the same operation)
}
#pragma offload_wait target(mic: 0) wait(in1) // Merging with MIC
```

the CPU can continue to perform the related calculation without waiting for the end of the MIC operation. It will not stop the calculation until it encounters a “wait” statement. Also, the CPU waits for the “signal” statement to be transmitted before proceeding with execution.

In the transition calculation module, the areas where the CPU and MIC collaborate mainly focus on the nested loops of the energy function; that is, they sum the potential energy of all atoms in the molecular system. For each calculation, the results from the previous calculation with the position and velocity of the atom are required. These data can be transferred from the CPU to the MIC card when CPU handles other tasks. As a result, the delay caused by data transmission is hidden within the CPU calculation cycle.

4 Experiment and Result Analysis

The experiments in this paper were carried out on the Tianhe-2 supercomputer. It has 16 000 computing nodes, each with two 12-core Xeon E5 processors and three 57-core Xeon Phi coprocessors (computing accelerator cards). It has a total of 32 000 Xeon E5 main processors and 48 000 Xeon Phi coprocessors; thus, a total of 312 000 computing cores. In 2012, it was ranked the first in the top 500 world's fastest supercomputers, with a peak computing speed of 54.9 Pflops (peta floating point operations per second). Currently, it is ranked the fourth[22]. The test environment for this paper is shown in Table 5.

4.1 Performance test and result analysis of the Crystal-KMC after optimization (using only the master cores of Tianhe-2)

(1) Small scale test

We use the optimized Crystal-KMC software to simulate the diffusion of vacancies in iron. The size of the simulated space is 200×200×200 and the number of particles is 1.6×10^7, among which the number of vacancies is 200. Furthermore, the concentration of copper atoms is 1%, the cutoff radius is 4, the temperature is 600 ℃, and the potential function is the EAM potential. The advance time of each step of the KMC simulation system is 1.25×10^{-8} s, for a single thread.

In this paper, Crystal-KMC is executed by a single process on each node using a different number of nodes (less than 128 nodes) each time. The execution results are then recorded, as shown in Table 6. The approach was executed three times in each case, and then we took the average as the final result. The unit of execution time is second (s). The test results of different scale tests are shown in Figs. 5 and 6.

(2) Large scale test

The simulated space size is 1000×1000×1000 and the number of particles is 1.6×10^7, among which the number of vacancies is 1000. Also, the concentration of the copper atom is 1%, the cutoff radius is 4, the temperature is 600 ℃, and the EAM potential function is used. Each step of the KMC simulation system has a forward time of 1.25×10^{-8} s for a single thread.

In this paper, Crystal-KMC is executed by a single process on each node using a different number of nodes (more than 128 nodes) each time. The execution results are recorded and counted, as shown in Table 7. The approach was executed three times in each case, and then we took the average as the final result. The execution time is in second (s).

The test results of different scale tests in Figs. 7 and 8 show that the Crystal-KMC improves the parallel execution performance of the original program after

Table 5 Specific configuration of Tianhe-2.

Configuration	Specific details
Operate system	Kylin Linux
MPI	MPI 3.0
CPU	32000 Intel Xeon CPU, 24 cores
Coprocessor	48 000 Intel Xeon Phi
Computing node	16 000
Network topology	TH Express-2
Memory	1.4 PB
Hard disk	12.4 PB

Table 6 Optimized Crystal-KMC parallel execution time (small-scale test).

Number of processes	Execution time (s)			Average execution time (s)
	No. 1	No. 2	No. 3	
1	1600.2	1591.4	1598.8	1596.8
2	836.7	849.9	850.8	845.8
4	457.6	461.1	459.8	459.5
8	266.4	272.6	259.2	269.4
16	157.8	152.5	153.5	154.6
32	83.1	87.2	85.3	85.2
64	49.4	51.2	51.5	50.7
128	30.0	33.1	30.5	31.2

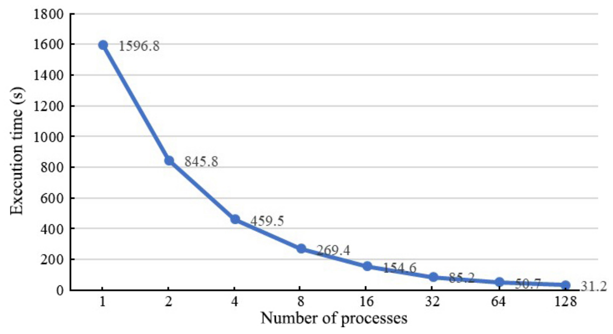


Fig. 5 Execution time of optimized Crystal-KMC (small-scale test).

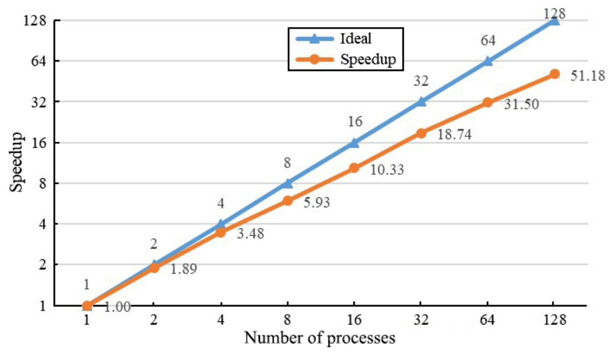


Fig. 6 Speedup of optimized Crystal-KMC (small-scale test).

Table 7 Optimized Crystal-KMC parallel execution time (large-scale test).

Number of processes	Execution time (s)			Average execution time (s)
	No. 1	No. 2	No. 3	
128	973.1	982.3	980.1	978.56
256	657.9	663.1	662.6	661.2
512	294.5	297.4	302.9	298.3
1024	196.7	190.8	190.3	192.5
2048	154.3	150.2	152.7	152.4

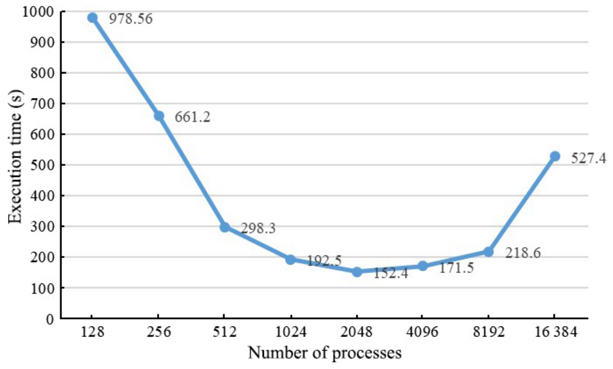


Fig. 7 Execution time of optimized Crystal-KMC (large-scale test).

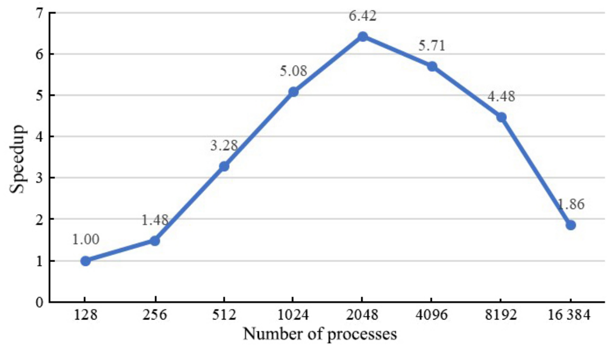


Fig. 8 Speedup of optimized Crystal-KMC (large-scale test).

master cores migration by the MPI. However, when conducting large-scale tests, as the number of processes increases, the parallel speedup decreases. The reasons are as follows:

The Crystal-KMC uses spatial region partitioning for parallel execution and the MPI message driver to reduce communication latency. However, the program is responsible for task scheduling and load balancing the main process when performing task partitioning. When the number of computing nodes is small, a single main process can complete task partitioning and load balancing tasks in time. However, when the Crystal-KMC uses more nodes to execute in parallel on the Tianhe-2 supercomputer, the amount of computation and communication required for task partitioning and load balancing will become larger. Exceeding the computing power of a single control process will lead to task congestion. As a result, load balancing will take more time and becomes a program acceleration bottleneck. In the actual test process, as the size of the node increases, the growth of the program speedup gradually decreases. But when the number of nodes reaches or exceeds 4096, the total execution time of the Crystal-KMC starts to increase.

4.2 Performance test and result analysis of the simulation module in the Crystal-KMC after optimization (using only the slave cores of Tianhe-2)

In this paper, we use OpenMP to optimize the Crystal-KMC software in parallel on the Tianhe-2 slave cores. Because the simulate module is the performance bottleneck in Crystal-KMC, we use offload mode to realize parallel optimization on the Tianhe-2 slave cores, which further improves its performance.

The simulated space is 200$\times$200$\times$200 and the number of particles is 1.6×10^7, among which there are 500 vacancies. The copper atom concentration is 1%, the truncation radius is 4, the temperature is 600 ℃, and the potential function is EAM potential. The advance time of each KMC simulation system is 1.25×10^{-8} s. To test the effect of using the offload mode, only the execution time of the simulate module is counted. The test results are shown in Figs. 9 and 10.

From Figs. 9 and 10, after optimization, the simulation module in Crystal-KMC obtains a higher parallel speedup when running on the slave cores.

4.3 Performance test and result analysis of the Crystal-KMC after optimization (master/slave collaborative cores parallelism of Tianhe-2)

The hybrid parallelism of the Crystal-KMC software is

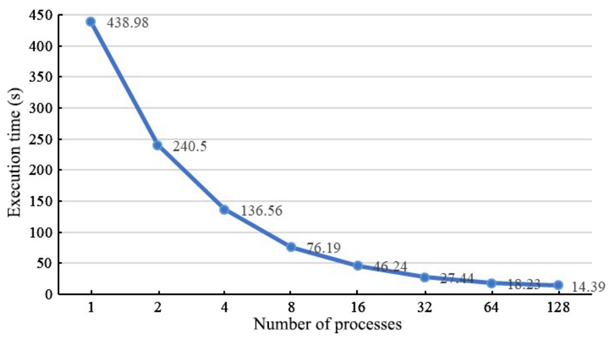


Fig. 9 Execution time of the simulation module in Crystal-KMC after optimization.

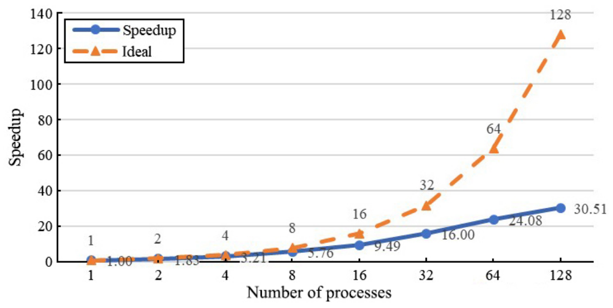


Fig. 10 Speedup of the simulation module in Crystal-KMC after optimization.

realized by using the master/slave collaborative cores parallelism of the Tianhe-2. The simulated space is 200×200×200 and the number of particles is 1.6×10^7, among which are 500 vacancies. The copper atom concentration is 1%, the truncation radius is 4, the temperature is 600℃, and the potential function is the EAM potential. The advance time of each KMC simulation system is 1.25×10^{-8} s. Table 8 shows the optimized Crystal-KMC parallel execution time (master/slave collaborative cores parallelism of Tianhe-2). The approach was executed three times in each case, and then we took the average as the final result. The unit of execution time is second (s).

From Figs. 11 and 12, when using the master/slave collaborative cores optimization, the overall execution time of the Crystal-KMC decreases while the speedup increases. In comparison to using only the master cores of Tianhe-2 (Fig. 5), the execution time reduces significantly for less than 32 cores. However, when the processes used more than 32 cores, the speedup increased slowly, because the serial portion of the program is not negatively affected by increments in thread number. When the scale of the simulated atom is larger, the advantage of multi-thread acceleration is more evident.

Table 8 Execution time of the optimized Crystal-KMC (master/slave collaborative cores parallelism of Tianhe-2).

Number of processes	Execution time (s)			Average execution time (s)
	No. 1	No. 2	No. 3	
1	485.45	488.39	489.44	487.76
2	292.13	288.30	290.62	290.35
4	184.34	187.20	194.59	188.71
8	126.09	130.31	132.52	129.64
16	95.13	96.16	95.39	95.56
32	81.38	80.42	76.94	79.58
64	73.30	72.47	71.28	72.35
128	64.88	66.13	65.88	65.63

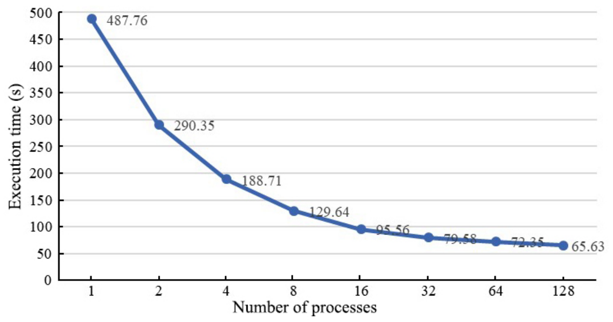


Fig. 11 Execution time of the optimized Crystal-KMC (master/slave collaborative cores parallelism of Tianhe-2).

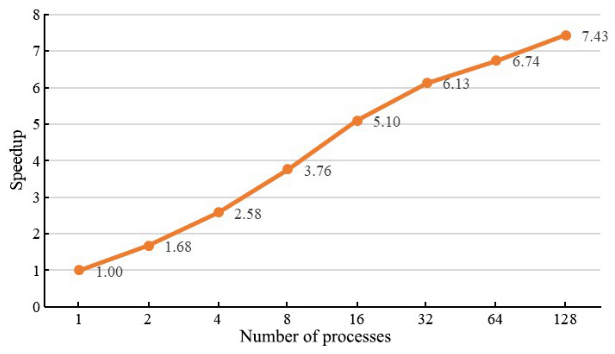


Fig. 12 Speedup of the optimized Crystal-KMC (master/slave collaborative cores parallelism of Tianhe-2).

5 Conclusion

To improve the scale of material irradiation damage simulation and make full use of the huge computing power of modern supercomputers, we propose the parallel optimization of Crystal-KMC on Tianhe-2. The computation-intensive modules of Crystal-KMC run on the MIC coprocessor. The multi-thread parallel optimization of Crystal-KMC is performed by OpenMP. Combined with existing MPI inter-process communication optimization, a hybrid parallel optimization is developed. The test results show that in the CPU+MIC collaborative parallel mode of Tianhe-2, the speedup of calculation hotspot in Crystal-KMC can reach up to 30.1 and the overall speedup of the software is 7.43.

Acknowledgment

This work was supported by the National Key R&D Program of China (No. 2017YFB0202104).

References

[1] Y. L. He, X. Y. Li, and Y. Wang, Current research status and progress of nuclear energy virtual reactor in China and abroad, (in Chinese), *Nucl. Sci. Technol.*, vol. 3, no. 2, pp. 41–47, 2015.

[2] A. F. Voter, Introduction to the kinetic Monte Carlo method, in *Radiation Effects in Solids*, K. E. Sickafus, E. A. Kotomin, and B. P. Uberuaga, eds. Dordrecht, Netherlands: Springer, 2007, pp. 1–23.

[3] M. S. Daw and M. I. Baskes, Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals, *Phys. Rev. B*, vol. 29, no. 12, pp. 6443–6453, 1984.

[4] M. J. Hoffmann, S. Matera, and K. Reuter, kmos: A lattice kinetic Monte Carlo framework, *Comput. Phys. Commun.*, vol. 185, no. 7, pp. 2138–2150, 2014.

[5] C. Domain, C. S. Becquart, and L. Malerba, Simulation of radiation damage in Fe alloys: An object kinetic Monte Carlo approach, *J. Nucl. Mater*, vol. 335, no. 1, pp. 121–145, 2004.

[6] V. A. Borodin, P. V. Vladimirov, and A. Möslang, Lattice kinetic Monte-Carlo modelling of helium-vacancy cluster formation in bcc iron, *J. Nucl. Mater.*, vol. 367, pp. 286–291, 2007.

[7] Y. Shim and J. G. Amar, Rigorous synchronous relaxation algorithm for parallel kinetic Monte Carlo simulations of thin film growth, *Phys. Rev. B*, vol. 71, no. 11, p. 115436, 2005.

[8] E. Martínez, J. Marian, M. H. Kalos, and J. M. Perlado, Synchronous parallel kinetic Monte Carlo for continuum diffusion-reaction systems, *J. Comput. Phys.*, vol. 227, no. 8, pp. 3804–3823, 2008.

[9] SPPARKS kinetic Monte Carlo simulator, http://spparks.sandia.gov/, 2019.

[10] X. Y. Liu, J. B. Adams, F. Ercolessi, and J. A. Moriarty, EAM potential for magnesium from quantum mechanical forces, *Model. Simul. Mater. Sci. Eng.*, vol. 4, no. 3, pp. 293–303, 1996.

[11] I. Martin-Bragado, A. Rivera, G. Valles, J. L. Gomez-Selles, and M. J. Caturla, MMonCa: An object kinetic Monte Carlo simulator for damage irradiation evolution and defect diffusion, *Comput. Phys. Commun.*, vol. 184, no. 12, pp. 2703–2710, 2013.

[12] M. Leetmaa and N. V. Skorodumova, KMCLib: A general framework for lattice Kinetic Monte Carlo (KMC) simulations, *Comput. Phys. Commun.*, vol. 185, no. 9, pp. 2340–2349, 2014.

[13] J. J. Li, P. Wei, S. F. Yang, J. Wu, P. Liu, and X. F. He, Crystal-KMC: Parallel software for lattice dynamics Monte Carlo simulation of metal materials, *Tsinghua Sci. Technol.*, vol. 23, no. 4, pp. 501–510, 2018.

[14] M. A. Bezerra, R. E. Santelli, E. P. Oliveira, L. S. Villar, and L. A. Escaleira, Response Surface Methodology (RSM) as a tool for optimization in analytical chemistry, *Talanta*, vol. 76, no. 5, pp. 965–977, 2008.

[15] C. C. Battaile, The kinetic Monte Carlo method: Foundation, implementation, and application, *Comput. Methods Appl. Mech. Eng.*, vol. 197, nos. 41 & 42, pp. 3386–3398, 2008.

[16] G. F. Xie, D. W. Wang, and C. T. Ying, Molecular dynamics simulation of Gd adatom diffusion on Cu(110) surface, (in Chinese), *Acta Phys. Sin.*, vol. 52, no. 9, pp. 2254–2258, 2003.

[17] C. Ruan, X. M. Sun, and Y. X. Song, Cellular method combined with Monte Carlo method to simulate the thin film growth processes, (in Chinese), *Acta Phys. Sin.*, vol. 64, no. 3, p. 038201, 2016.

[18] L. G. Wang and P. Clancy, Kinetic Monte Carlo simulation of the growth of polycrystalline Cu films, *Surf. Sci.*, vol. 473, nos. 1 & 2, pp. 25–38, 2001.

[19] Intel VTune^{TM} Amplifier XE, https://software.intel.com/en-us/vtune, 2018.

[20] B. Shen, G. Y. Zhang, S. H. Wu, X. W. Lu, and Q. Zhang, Research of offload parallel method based on MIC platform, (in Chinese), *Comput. Sci.*, vol. 41, no. 6A, pp. 477–480, 2014.

[21] X. Liu, Research on parallel acceleration technology for AMBER: A molecular dynamic simulation software on TianHe-2 supercomputer, (in Chinese), Master dissertation, National University of Defense Technology, Changsha, China, 2015.

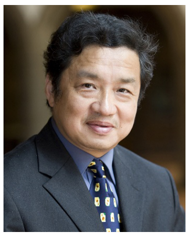
[22] China extends lead in number of TOP500 supercomputers, US holds on to performance advantage, https://www.top500.org/, 2019.

Jianjiang Li is currently a professor at University of Science and Technology Beijing, China. He received the PhD degree in computer science and technology from Tsinghua University in 2005. He was a visiting scholar at Temple University from Jan. 2014 to Jan. 2015. His current research interests include parallel computing, cloud computing, parallel compilation, and big data.

Yun Yang is currently a master degree candidate at University of Science and Technology Beijing, China. She received the bachelor degree from Taiyuan University of Technology in 2017. Her current research interests include parallel computing and cloud computing.

Baixue Ji is currently a master student at University of Science and Technology Beijing, China. She received the BS degree from South-Central University for Nationalities in 2018. Her current research interests include parallel computing and big data.

Peng Wei received the master degree from University of Science and Technology Beijing, China in 2019. He is currently a development engineer at Industrial and Commercial Bank of China Shangdong Branch. His research interests include parallel computing, cloud computing, and big data.



Jie Wu received the PhD degree from Florida Atlantic University in 1989. He serves as the director of Center for Networked Computing and Laura H. Carnell professor at Temple University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He is a fellow of IEEE.