TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 06/12 pp296–308
DOI: 10.26599/TST.2019.9010074
Volume 26, Number 3, June 2021

Memory Access Optimization of Molecular Dynamics Simulation Software Crystal-MD on Sunway TaihuLight

Jianjiang Li, Jie Lin, Panpan Du*, Kai Zhang, and Jie Wu

Abstract: The radiation damage effect of key structural materials is one of the main research subjects of the numerical reactor. From the perspective of experimental safety and feasibility, Molecular Dynamics (MD) in the materials field is an ideal method for simulating the radiation damage of structural materials. The Crystal-MD represents a massive parallel MD simulation software based on the key material characteristics of reactors. Compared with the Large-scale Atomic/Molecurlar Massively Parallel Simulator (LAMMPS) and ITAP Molecular Dynamics (IMD) software, the Crystal-MD reduces the memory required for software operation to a certain extent, but it is very time-consuming. Moreover, the calculation results of the Crystal-MD have large deviations, and there are also some problems, such as memory limitation and frequent communication during its migration and optimization. In this paper, in order to solve the above problems, the memory access mode of the Crystal-MD software is studied. Based on the memory access mode, a memory access optimization strategy is proposed for a unique architecture of China's supercomputer Sunway TaihuLight. The proposed optimization strategy is verified by the experiments, and experimental results show that the running speed of the Crystal-MD is increased significantly by using the proposed optimization strategy.

Key words: molecular dynamics simulation; Crystal-MD; Sunway TaihuLight; memory access optimization

1 Introduction

After the Molecular Dynamics (MD) was invented in the 1950s[1], it has gradually attracted wide attention. Molecular dynamics simulates the molecular trajectory by solving the molecular motion equations of a system, and obtains the macroscopic properties of the system through the description of some microscopic quantities, such as atomic velocity, momentum, potential energy, and others, and the application of statistical methods on them. By using the molecular dynamics simulation softwares to simulate the molecular trajectory, not only the molecular motion scene can be restored more realistically, but also the experimental time can be reduced significantly. Currently, the most popular molecular dynamics simulation softwares are the Large-scale Atomic/Molecurlar Massively Parallel Simulator (LAMMPS) software[2] developed by the Sandia National Laboratory of the United States and the ITAP Molecular Dynamics (IMD) software[3] developed by University of Stuttgart of Germany. However, both of them store a large amount of neighbor atomic index data, requiring a large amount of memory space, which poses great challenges to the simulation of a larger molecular scale. In order to overcome this memory-

• Jianjiang Li and Jie Lin are with the Department of Computer Science and Technology, University of Science and Technology Beijing, Beijing 100083, China. E-mail: lijianjiang@ustb.edu.cn; jaelyn_lin@163.com.
• Panpan Du is with Dawning Information Industry Co., Ltd., Tianjin 300384, China. E-mail: PanDuYee@163.com.
• Kai Zhang is with Beijing Sogou Technology Development Co., Ltd., Beijing 100084, China. E-mail: zhchhz@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

related problem, Bai et al.[4] proposed a new data structure named the Lattice Neighbor List based on the characteristics of the Body-Centered Cubic (BCC) structural metals to reduce the memory space occupied by neighbors' atomic information. The communication model of parallel MD simulation was designed for this data structure proposed in Ref. [4]. Based on this data structure and a communication model, a large-scale parallel MD simulation software Crystal-MD[5] was developed, which greatly reduced the memory space required for software operation. The core algorithm of the Crystal-MD software is based on the MD simulation of multithreading technology optimization, and a partition step calculation method, namely, the partition two-step method, is employed to avoid write conflicts and improve the parallelism of thread processing. Recently, the Crystal-MD software has been implemented in Sunway TaihuLight and optimized to a certain extent. With the aim to further improve the performance of this software, this paper studies the Crystal-MD memory access optimization of the unique architecture of Sunway TaihuLight and proposes a new optimization strategy for the memory access of the salve core.

The main contributions of this paper are as follows:

(1) This paper summarizes the optimization methods of the memory access of the slave core of Sunway TaihuLight.

(2) This paper introduces the basic principle of the Crystal-MD software and parallel techniques, and points out the shortcomings of the traditional Crystal-MD software in the calculation of the Embedded-Atom Method (EAM) potential and the corresponding force that is expressed as computeEam function.

(3) Aiming at the memory hierarchy of Sunway TaihuLight, this paper divides the limited Local Data Memory (LDM) of the slave core into two parts, Software Cache (SWC) and Direct Access Cache (DAC), and determines the access mode according to the access characteristics of the variables used in the Crystal-MD software. In the optimization based on the access mode, the double-buffering strategy is used for DAC, while for SWC, a strategy of simulating the function of the cache is applied to get data.

(4) The performance test of Crystal-MD software before and after the optimization on Sunway TaihuLight platform is conducted.

The rest of this paper is organized as follows. The storage mode of Crystal-MD software on Sunway TaihuLight platform is analyzed and optimized in Section 2. The experimental results and analysis are provided in Section 3. The research status of the relevant work is introduced in Section 4 and the conclusions are summarized in Section 5.

2 Memory Access Optimization of Crystal-MD on Sunway TaihuLight Platform

2.1 Introduction to Sunway TaihuLight

Sunway TaihuLight represents China's first supercomputer built entirely with domestic processors. It is world's the first supercomputer with a peak speed of more than 100 Pflops (peta floating point operations per second). Currently, it is deployed in the National Supercomputing Center of Wuxi in China. The peak speed of Sunway TaihuLight is 125.4 Pflops, and it has a continuous operation speed of 93 Pflops. Its power consumption is 6.05 billion times per watt, and it includes a total of 40 960 SW26010[6] heterogeneous multi-core processors developed by the National High Performance Integrated Circuit Design Center with its own core technology.

In the SW26010 heterogeneous multi-core processor, each slave core has a 64 KB Scratch Pad Memory (SPM) that can be controlled by the user as an LDM. The SPM can be configured to support fast buffers with a precise user control or software emulation caches that implement automatic data caching. Because of a poor performance of the latter, SPM is mostly used in the form of the former. The core directly accesses the register to obtain the data, and after the calculation is finished, the result is sent back to the register, as shown in Fig. 1. The register can read data in the LDM or directly communicate with the main memory to obtain the data. The LDM can transfer data in bulk with main memory through the Direct Memory Access (DMA). According to Ref. [6], it takes 177 clock cycles to read the main memory

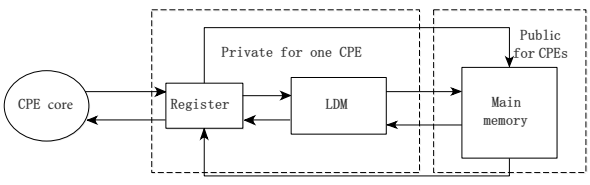


Fig. 1 Storage hierarchy of Computing Processing Element (CPE) in the SW26010 processor.

data directly from the slave core through the register, while it takes only four clock cycles to read the data from the register in the LDM. Therefore, it has been a common method to transfer the required data to the LDM in bulk in advance, thus achieving better acceleration effect. However, the storage space of the LDM is only 64 KB, so an effective task division is required to balance the calculation amount allocated to each slave core. Moreover, it is crucial to make good use of the multi-layer storage architecture of the slave core to transfer the data needed for the current-stage calculation in the LDM, so as to reduce the number of accesses to the slave core memory.

The Sunway TaihuLight currently ranks the third in the latest Top500 list, and it has attracted extensive attention from many scholars and experts. Some of the recent achievements in the research on the storage optimization of Sunway TaihuLight are presented in Table 1.

There are also analysis and optimization of multi-core architecture in Refs. [13–15].

2.2 Introduction to traditional Crystal-MD parallel software

2.2.1 Basic principles of MD method

Molecular dynamics represents a molecular simulation method based on the classical Newton laws of motion. This method firstly calculates the potential energy of an atom by using the potential energy function, which is generally the empirical function, according to the initial position of the atom in the system, and then determines the interaction force on the atom. Finally, according to Newton's second law, which is given by $F = m \times a$, the atom acceleration can be obtained. Based on the above steps, Newton's equations can be integrated over time to obtain the velocity and position of the system's initial state, and the iterative process can be carried out step by step to obtain the position and velocity of atoms at any time. The flowchart of a general molecular dynamics simulation is presented in Fig. 2. NVT is a shorthand for canonical ensemble, which represents that the number of particles (N), volume (V), and temperature (T) are certain. NVE is a shorthand for micro-canonical ensemble, which represents that the number of particles (N), volume (V), and the total energy (E) are certain. In the selection of potential function, Modified Embedded Atom Method (MEAM) potential is an extension of EAM potential. It introduces an angle factor to describe the interaction between atoms more accurately, but the fitting is complicated.

In this paper, the EAM potential is used as an example to introduce the MD parallel algorithm. The EAM consists of a typical pair of potentials plus an interbody embedding potential. The EAM[16] can be expressed as

$$E_{\text{tot}} = \sum_{i}^{n} e_i + \sum_{i}^{n} G_i(\rho_i) \tag{1}$$

Table 1 Research on memory access optimization of Sunway TaihuLight.

Targeted problem	Solution	Project proposer
High access delay caused by irregular access.	Data prefetching using asynchronous DMA.	Hunan University (Dong et al.[7])
The limited memory bandwidth does not match the powerful computing power of Sunway TaihuLight platform.	Partition the local memory according to the characteristics of the computing core and perform data sharing between the cores through register communication.	State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences (Yang et al.[8])
The slave core's memory space is small, and the program has a large number of discrete memory accesses.	Adjust the storage order of the original discrete array in order to facilitate communication, and then the core performs the communication operations of reading data, calculating and writing back data, and finally calculating the data that the core writes back to the original storage order again.	Hunan University (Hong et al.[9])
Matrix multiplication optimization based on double-precision universal format of SW26010 multi-core processor.	A block algorithm at three levels of memory, SPM, and registers is proposed to coordinate data on the memory hierarchy.	Institute of Software, Chinese Academy of Sciences (Jiang et al.[10])
The cores of the same core group share data through main memory, which has a large overhead.	Use registers communication to design a data transfer protocol from the core so that data can be shared without main memory.	Chinese Academy of Sciences (Ao et al.[11])
Data accessed irregularly from memory transfer causes serious bandwidth occupation.	Use software Cache to store data accessed irregularly, design appropriate data structures for applications, reduce data reuse distance, and improve reusability.	University of Science and Technology of China (Yu et al.[12])

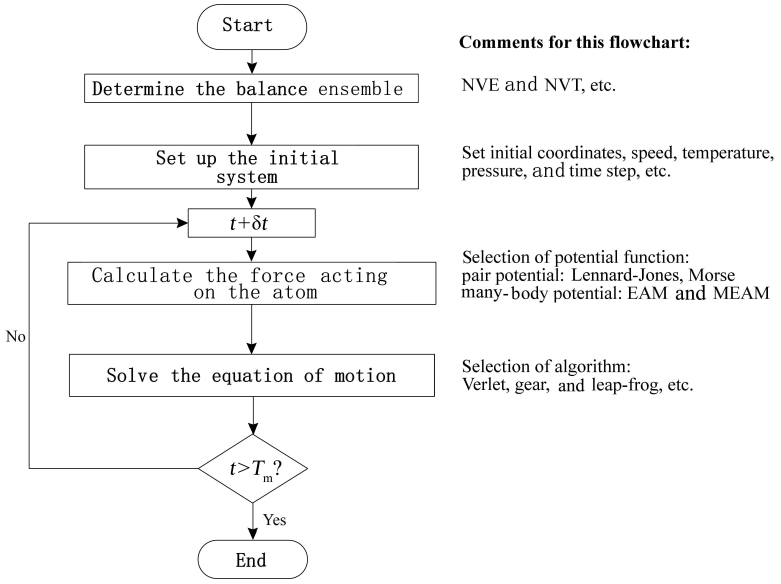


Fig. 2 Flowchart of a general molecular dynamics simulation.

where e denotes the pair potential, G denotes the embedding energy, and the electron density ρ represents the sum of the actions of surrounding atoms. The e_i and ρ_i are expressed as

$$e_i = \frac{1}{2} \sum_{j=1, j\neq i}^{n} \Phi_{ij}(r_{ij}) \tag{2}$$

$$\rho_i = \sum_{j=1, j\neq i}^{n} f_j(r_{ij}) \tag{3}$$

where r_{ij} denotes the distance between atoms i and j, Φ_{ij} denotes the two-bady pair potential between atoms i and j, and f_j denotes the electron density generated by atom j.

The MD simulation has the following characteristics:

(1) Both the number of particles and the number of time steps in the MD simulation system are very large, so the calculation amount is huge.

(2) Particle computing information needs to synchronize at each step, resulting in a huge amount of communication.

A general block diagram of the MD simulation parallel computing method is presented in Fig. 3. The Crystal-MD parallel program uses the Spatial-Decomposition (SD)[17, 18] method to distribute the actual simulated spatial physics evenly to each node machine such that each node machine calculates only the force of the particles in its subdomain. Thus, when particles move from one subdomain to another, the particles are exchanged between the nodes.

2.2.2 Traditional Crystal-MD parallel program of Sunway TaihuLight

The large-scale parallel MD simulation software Crystal-MD aims at the structural characteristics of the BCC structural metals, and the data structure is the Lattice Neighbor List[4].

According to the spatial distribution order of atoms, the Lattice Neighbor List stores the position, speed, and other related information of atoms in the specific positions of corresponding arrays one by one so that the array index corresponds to the Lattice points in the space one by one. Based on the truncation radius and lattice constant, the program calculates the offset of the index of a center atom from the index of the neighbor atom in an array. When an atom runs away from its original lattice point, the program allocates extra memory space to store its information. Based on the characteristics of the BCC structure of a metal, the atoms are substantially fixed at the lattice points throughout the simulation. Therefore, at the beginning of the simulation, the program calculates the indexes of the atoms that need to be sent to and received from the neighbor process, and stores these indexes in an array. In each subsequent time interval, atom information are sent and receivced according to the index stored in the array. The Crystal-MD divides the simulation area into several blocks by using the two-step method and divides the simulation blocks into two groups such that blocks within one group are not adjacent to each other, and then calculates the potential energy and force between atoms in the two groups, respectively, so as to avoid write conflicts between the threads.

2.2.3 Current problems in traditional Crystal-MD parallel programs

After conducting a series of statistics of the program

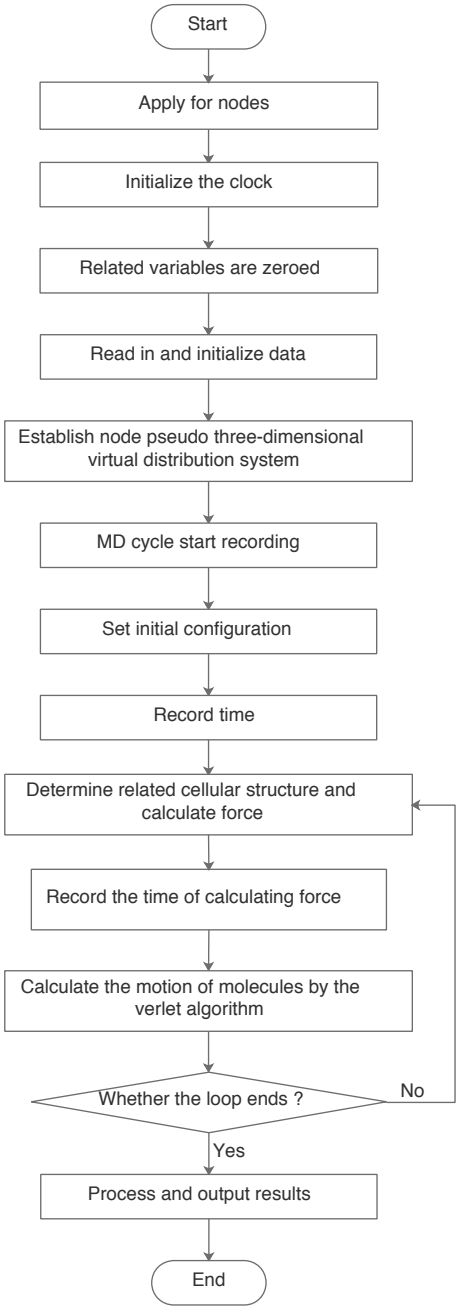


Fig. 3 General block diagram of the MD simulation parallel computing method.

running time, it is found that the EAM potential and its corresponding force calculation, which is expressed as computeEam function in the program, are the most time-consuming tasks in Crystal-MD program. The calculation process can be divided into the following three stages. In the first stage, the pair potential energy between atoms is calculated, and the density of the electron cloud is obtained by interpolating the distance between the atoms using the interpolation function. In the second stage, the embedding energy and its derivative are calculated. In the third stage, the interaction between the atoms is calculated by embedding the energy and potential energy. In these three stages, during the calculation, data information on the potential table are required; thus, the table lookup based on the atomic distance is required during the calculation process. The potential table size is about 280 KB, but as the storage capacity of the LDM is only 64 KB, it is impossible to load the entire potential table into the LDM all at once.

The Crystal-MD program adopts the method of storing partial data and obtaining the rest data by interpolation to solve the mentioned problem. Namely, in the program initialization stage, 5000 values are read from the potential table file, used as the initial value (about 40 KB), and used for cubic spline interpolation, while 35 000 values are inserted for later calculation. This initial incoming data represent static data in the LDM and are resident in them; thus, in the initial phase of the calculation, only one DMA transfer is needed, which significantly reduces the amount of data transferred to the potential table.

There are many nested loops in the computeEam function, which are loaded onto numerous slave cores by the Crystal-MD program for further computation through an interface in the athread library provided by the platform. As shown in Fig. 4, the data accessed by rules are transferred between the main memory and local memory through the DMA in the cyclic calculation process, while the potential table, as static data, is put into local memory at once before the cycle starts.

However, there are two obvious drawbacks of the above method:

(1) Most data stored in the potential table are obtained by interpolation, so there may be deviations from the original table, and when the deviation is amplified, it will have a significant impact on the final result. Therefore, the complete data stored in the potential table need to be calculated, which increase the calculation amount significantly.

(2) 5000 double data are resident in the LDM, causing

```
1 do k=zstart, zend
2    do j=ystart, yend
3       DMA_GET;          //Read data from main memory
4       do i=1, m
5             ...
6       enddo
7       DMA_PUT;          //The result of the calculation is written back to the main memory
8    enddo
9 enddo
```

Fig. 4 Pseudo-code of the calculation of slave core data.

that the LDM space of 40 KB, which is about 5/8 of the LDM total capacity, is occupied all the time. Therefore, other regular-access variables can only reduce the data granularity per transfer into the LDM, which puts more pressure on the LDM storage space that is of a small capacity.

2.3 Crystal-MD optimization strategy for slave core memory access on Sunway TaihuLight

2.3.1 Memory access analysis of the main variables of the Crystal-MD

With the aim to overcome all the above-mentioned problems, considering the specific structure of Sunway TaihuLight platform, this paper suggests optimizing the three stages of the EAM potential and the corresponding force calculation. However, due to the limited space of paper, only one of these phases, the calculation of potential energy between the atoms, is presented. The computational overhead of this phase is related mostly to the nested loops, so that it should be optimized. The amount of computation varies with the size of the simulated atom and the number of the processes. The main variables involved in the optimization process are the x-array of three-dimensional position information of atoms, the rho array of the electron cloud density information of atoms, and the rho_spline array of the potential energy table of the electron cloud density. It is worth noting that the rho_spline array of the potential table is complete, and its size is about 280 KB. Arrays x and rho are acquired and calculated in turn according to the subscripts at the programming time, so they belong to the regular access memory. However, from the perspective of variable size, even with the small number of atomic simulations and more calculations from the kernel, the size of each array variable allocated from the kernel is likely to exceed the LDM capacity size of the kernel (64 KB).

Array rho_spline of the electron cloud density and potential energy is a static table, with the size of 280 KB, which is used to look up the table according to the atomic distances calculated by the atomic potential energy calculation process and obtain the electron cloud density and potential energy of the corresponding distances for subsequent calculation. Therefore, it belongs to the irregular access memory. However, in the calculation process, the access of array rho_spline is also local. After a certain period of program running, the continuous access records of array rho_spline are counted, as shown in Fig. 5. In Fig. 5, the abscissa denotes the sequence

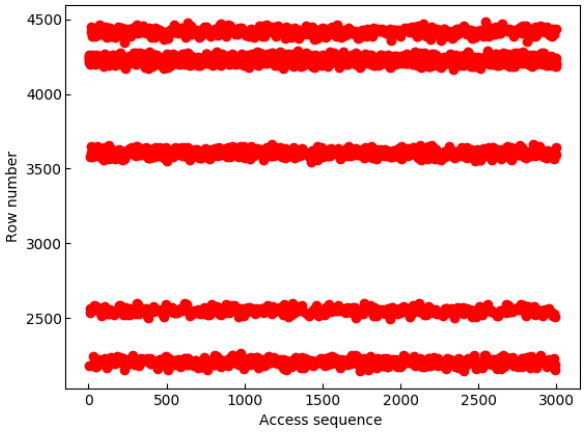


Fig. 5 Statistics of 3000 consecutive access records of rho_spline array.

of the nuclear access array and the ordinate denotes the row number of the nuclear access array. The array has only four columns, so the row number can represent the locality of array rho_spline to a certain extent. The red pattern in Fig. 5 is composed of the coordinate points located at the horizontal and vertical coordinates. As can be seen from Fig. 5, although the data access span is relatively large in the access records, the data access is generally concentrated around the line number of 5 categories, indicating that array rho_spline shows certain locality characteristic. Thus, from the storage perspective, arrays should be stored in the SWC, so that they can be fully reused.

Following the specific structure of Sunway TaihuLight platform, this paper divides the LDM storage space into SWC and DAC, which store different types of variables. Generally, the SWC simulates the function of the cache, which is suitable for storing and accessing irregular and reusable data, while DAC directly accesses data, which makes it suitable for storing regular data. According to the above analysis, arrays x and rho are suitable for storing in the DAC, while array rho_spline is suitable for storing in the SWC.

2.3.2 SWC implementation and optimization

In this work, the SWC is divided into an index section and a data section, where the data in a cache row correspond to the index information at the same offset of the index section, as shown in Fig. 6. In Fig. 6, hashs 1

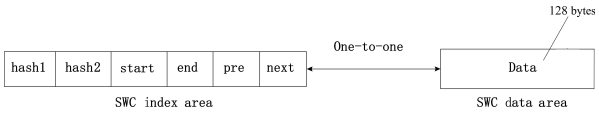


Fig. 6 Structure of a cache row.

and 2 denote the hash values calculated by hash functions 1 and 2, respectively, and they are used to determine whether the data belong to the current cache line. The parameters of both hash functions are array names of variables. The start field represents the subscript index value of the array of the first data of the corresponding row in the data area, and the end field represents the subscript index value of the array of the last data of the corresponding row in the data area. The start and end fields are used to determine the range of variables in the row. If the data are in the cache row, the index must be between the start and end fields. The pre field represents the row before the current row in the Least Recently Used (LRU) replacement policy, and the next field represents the row after the current row in the LRU replacement policy, which mainly records the most recent data access order and is used for data replacement.

The address mapping of the SWC adopts the hash mapping method, whose steps are as follows:

(1) Determine the hash function to be used for addressing. Set it as $\text{Hash}_{\text{offset}}(x)$, where x denotes the unique identifier name of an array element, such as a string consisting of the array name and the index value of array element.

(2) Mark the hash value obtained in Step 1, and modulo the cache capacity value (measured in the cache row) as the starting offset.

Consider the example of data reading from the SWC, suppose the calculation core needs to read an element Array$[i]$, and get the starting index through the address mapping. Assume that the cache row at the beginning of the index location includes the data, hash1 has a value of h_1, and hash2 has a value of h_2. Set the hash values of Array$[i]$ to be hash_cal_1 and hash_cal_2. If h_1 is equal to hash_cal_1 and h_2 is equal to hash_cal_2, the process continues; otherwise, it is determined that the data are not in the cache line. The start and end values from the index row are assumed to be v_1 and v_2, respectively. If i is in the interval $[v_1, v_2]$, then the data are stored in the cache row and Array$[i]$ can be quickly located to the specific location of the data in the cache row. Otherwise, the required data are not in the Cache line. If the data are not in the current cache row, the starting offset will be incremented by 1, and the desired data will continue to be queried. Thus, the program either exits the loop, when the desired data are found, or continues the loop until blank rows are encountered, ending the query process.

In this article, the first blank line found by a sequential fashion is called the most recent blank line and the corresponding index row is called the most recent index row. When the required data are not queried after the above-described process, it is concluded that the data have not been stored in the SWC. So it is needed to communicate with the main memory, obtain the needed data from the main memory, and store the obtained data in the SWC along with the adjacent data. The address mapping of the data storing process is the same as the address mapping of the data reading process, and since the data in both processes are identical, their starting indexes are the same. When the starting index is empty, the data index information are stored in the index row and the data are stored in the corresponding cache row of the data area. However, when the leading index row is not empty, data collision occurs, indicating that other data are also located to that row. By using the address mapping and conflict resolution, the nearest index row is located, the data index information are stored in it, and the data are stored in the corresponding cache row.

Accordingly, it can be concluded that empty lines play an important role in the SWC. Therefore, in this work, a relevant threshold f_z for the SWC cache row utilization u is set, requiring that $u \leqslant f_z$, so as to leave a certain number of empty lines. For a certain value of f_z, the addressing cost and data-replacement overhead of the data are determined based on the underlying architecture of Sunway TaihuLight and its specific parameters. The adaptive load factor f_z of the next stage is obtained using the sum of the minimized costs as the objective function, and finally, the addressing overhead and replacement overhead of the data are balanced. In this work, the granularity of the DMA transfers when the SWC communicates with the main memory is also optimized, and a new adaptive cache row optimization strategy is proposed. The cache hit ratio is calculated during the run of the program operation, and different sizes of cache rows are selected based on the cache hit ratio.

Suppose that the calculation of the core requires the element rho_spline$[i]$ in variable rho_spline. The process of reading rho_spline$[i]$ from the SWC is shown in Fig. 7.

2.3.3 DAC optimization

The DAC generally stores regular data, and in this work, the double-buffering strategy is employed to optimize the data transfer and computation. Among them, two buffers of the same size need to be used;

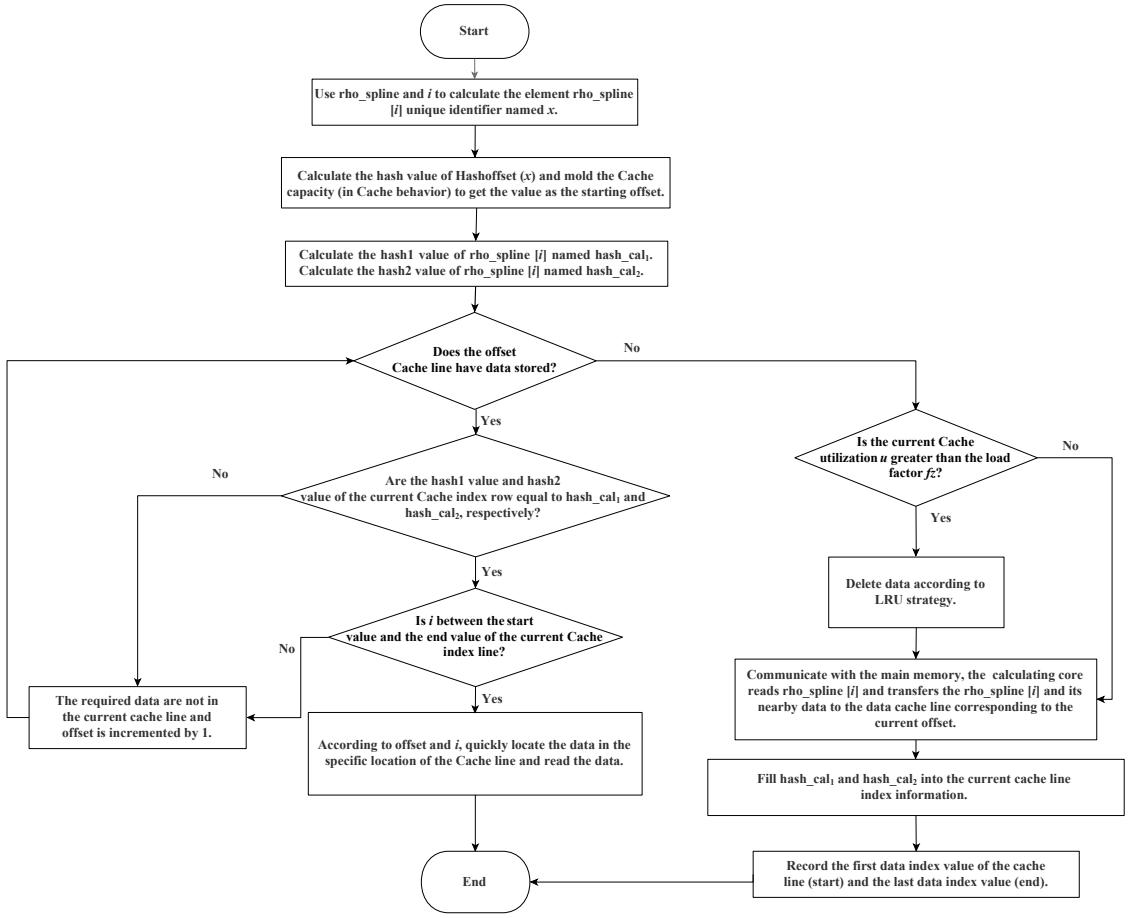


Fig. 7 Flowchart of the data reading process from the SWC.

for the convenience of illustration, the buffers are numbered as Nos. 1 and 2. No. 1 storage space of the current stage denotes the data-computation area, where the data needed for the current computing stage are stored. When the computational core needs the data, the data-computation area is accessed to obtain the required data. No. 2 storage space denotes the data-buffering area, which is responsible for receiving the next computing phase of data transferred from the main memory. In the next computing phase, the core gets the data from the No.2 storage space, so No.1 storage space becomes the data-buffering area. Because the DMA data transfer operation is asynchronous, the kernel can only send DMA instructions to the DMA controller and then return to continue the relevant operation; the rest of the data transfer is conducted by the DMA controller. Therefore, data computation and data transmission can be performed at the same time. When the communication cost is less than the computational cost, the slave core acceleration can achieve an ideal parallel acceleration effect.

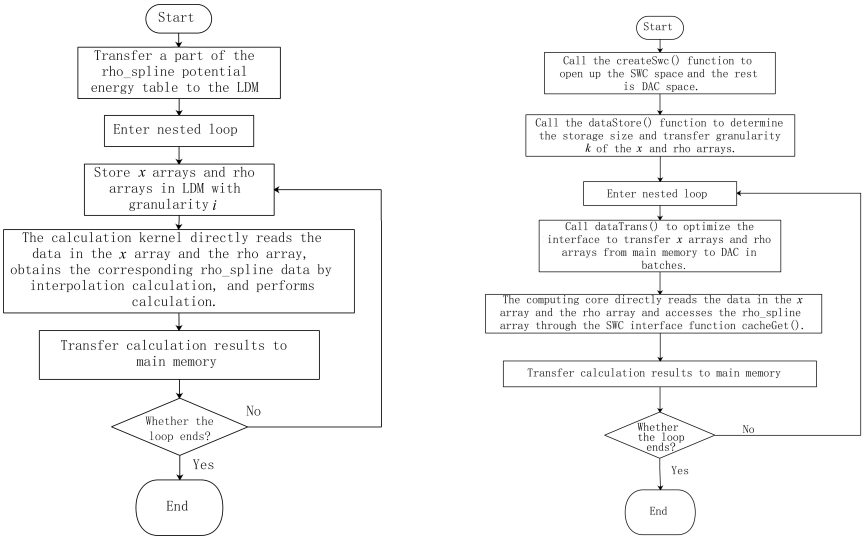
2.3.4 Implementation of optimized crystal-MD program

The comparison of the flowcharts of the original Crystal-MD program and the optimized Crystal-MD program is presented in Fig. 8.

Compared with the original Crystal-MD program, the optimized Crystal-MD program determines the storage mode of variables by the storage-optimization process proposed in this paper and then applies different optimization strategies to the variables with different storage modes. In the Crystal-MD optimization, the main optimization process includes the follwing steps:

(1) Use the createSwc() function to allocate index and data areas for the SWC in the LDM, where the underlying call from the core memory allocation function ldm_nalloc() returns pointers to the index and data areas. Then the remaining unallocated storage space of the LDM is used as the DAC.

(2) Call the dataStore() function to get the required storage space of arrays x and rho in the DAC and their transfer granularity, respectively. According to the data



(a) Flowchart of the original Crystal-MD program (b) Flowchart of the optimized Crystal-MD program

Fig. 8 Comparison of the flowcharts of the original Crystal-MD program and optimized Crystal-MD program.

storage strategy based on data access mode proposed in this paper, the dataStore() function models the incoming variable parameters, such as arrays x and rho. The model is essentially a linear constraint problem. After finding the solution to the model, the solution is returned as the size of storage space and the corresponding transmission granularity. In the original Crystal-MD program, after storing the 40 KB rh_spline potential table into the LDM, only 24 KB of the storage space is left for arrays x and rho. However, if the SWC capacity is set to 16 KB, the total space allocated for the data with regular memory access mode by dataStore() function will be 48 KB, so the rule data transfer granularity can be increased and the DMA transfer times can be reduced.

(3) Take the starting address of the storage space in the slave core, the starting address of the storage space in the main memory, and the DMA transfer granularity of arrays x and rho as parameters and pass them to dataTrans() function, and return the information on the data transfer success. Function dataTrans() encapsulates the details about data transfer, where the underlying DMA operation primitive described in Section 2.3.3 is used to transfer data in combination with the parameters passed in by the function dataTrans(). And the double buffer optimization strategy is encapsulated, which is convenient for users to call directly. Since the potential table is not stored as static data, a larger double buffer space can be allocated, which greatly improves the computational efficiency of regular data.

(4) Since the arrays x and rho have been transferred to the designated storage location in advance, it can be accessed directly according to its subscript index when calculating the nuclear access. Since the rho_spline potential table is stored in the SWC, it needs to be accessed through the interface function cacheGet(). Function cacheGet() encapsulates the process of computing that the core reads the data in the SWC. Firstly, it is determined whether the required data are in the SWC via a series of successive steps including address location and conflict resolution. If the required data have already been stored in the SWC at the time of computation, the data are retrieved; otherwise, the data are acquired via DMA communication from the main memory. Next, the data are sent to the calculation core, and then stored in the SWC with adjacent data, waiting for the next reuse.

3 Experimental Result and Analysis

3.1 Experimental environment and program

The test environment was the Sunway TaihuLight supercomputer, and the specific experimental parameters are provided in Table 2.

The test programs are the original Crystal-MD program[5], the traditionally optimized Crystal-MD program, which was optimized with the multi-way set-associative cache by our team, and the proposed optimized Crystal-MD program in this paper.

3.2 Test result and analysis of proposed optimization strategy

The running time of the original Crystal-MD program, the traditionally optimized Crystal-MD program, and the

Table 2 Experimental parameters.

Environment	Parameter	Remark or value
Hardware environment	Operating system	Sunway RaiseOS 2.0.5, Linux
	Processor	SW26010 multi-core processor
	CPU frequency (GHz)	1.45
	Main memory capacity per node (GB)	32
	Processor memory bandwidth (GB/s)	136.51
	Maximum number of nodes used	4000
	Maximum use from the number of cores	256 000
Software environment	Basic programming language	C, C++, and Fortran
	Parallel programming language	MPI 3.0, OpenMP 3.1, OpenACC 2.0, and Athread library
	Compiler	SWACC compiler

proposed optimized Crystal-MD program was tested in the small-scale (2×10^6 atoms), medium-scale (1.28×10^8 atoms), and large-scale (8.192×10^9 atoms) test environments, respectively.

The experimental results of the above three programs in a small-scale test environment are presented in Fig. 9. As can be seen in Fig. 9, under the same auditing, the order of the running time was the original Crystal-MD program, the traditionally optimized Crystal-MD program, and the proposed optimized Crystal-MD program, which showed that in the Crystal-MD programs, storing the non-regular access variables firstly into the software cache had a certain optimization effect. Moreover, the storage optimization strategy proposed in this paper resulted in better optimization effect than the traditional optimization strategy, and the running time was lower by 18.1% than that of the traditionally optimized Crystal-MD program, and by 30.8% than that of the original Crystal-MD program. The reasons for such results are as follows: The non-regular access array rho_spline had a certain locality, and the use of software cache could increase the reuse rate of data. Also, since rho_spline array was stored in the software cache, rather than in the LDM as static data, it could increase the granularity of each transfer of rule access data x and

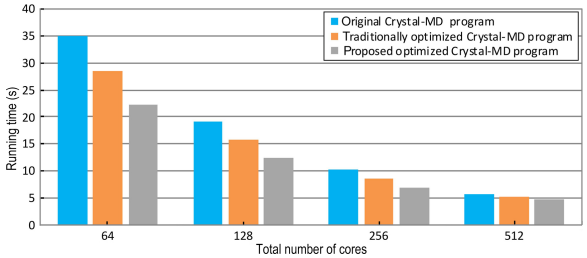


Fig. 9 Running time of the original Crystal-MD program, traditionally optimized Crystal-MD program, and proposed optimized Crystal-MD program in the small-scale test environment.

rho arrays and reduce the number of DMA transfers. In addition, the software cache designed in this article did not have a group-based concept and did not limit the mapping location of the cache rows, so the data replacement frequency was lower.

It is worth noting that the running time of the original Crystal-MD program declined faster with the number of cores grew than that of the other two programs. This was because when the number of cores increased, the amount of data allocated to each computational core decreased. In the original Crystal-MD program, rho_spline array was partially stored in the LDM, the rest was calculated by the interpolation process, and the number of DMA transfers reduced after the data amount of x and rho arrays to be calculated by each core reduced. The LDM was sufficient to store the arrays of x, rho, and rho_spline, etc., when the number of computed cores was large enough to require only an initial DMA transfer, so the running time of the original Crystal-MD program could be reduced faster by increasing the number of computed cores. In the optimized Crystal-MD programs that used the cache, increasing the transfer granularity of the rule access variable reduced the number of DMA transfers. However, when the computation amount of each core was reduced, the DMA transfer times of regular-access data of the traditionally optimized Crystal-MD program did not differ signigicantly from that of the original Crystal-MD program. However, multiple DMA transfers of irregular-access data were still needed, so further optimization could not be achieved. The original Crystal-MD program used the interpolation to calculate rho_spline array, and the final result had a certain error. Moreover, in the large-scale atomic simulation, the proposed optimized Crystal-MD program only needed a small number of cores to achieve excellent simulation effect, while the original Crystal-MD program needed 2–4 times larger number of cores to achieve a similar

simulation effect.

The experimental results of the above three programs in the medium-scale and large-scale test environments are presented in Figs. 10 and 11, respectively. Due to the constraints, the maximum number of used cores was 256 000. In the medium-scale test environment, the running time of the proposed optimized Crystal-MD program was 18.3% lower than that of the traditionally optimized Crystal-MD program, and 32.9% lower than that of the original Crystal-MD program. In the large-scale test environment, the running time of the proposed optimized Crystal-MD program was 19.6% lower than that of the traditionally optimized Crystal-MD program, and 33.9% lower than that of the original Crystal-MD program. In all the test environments, the proposed optimized Crystal-MD program had a good optimization effect, indicating that the access optimization strategy proposed in this paper provided good scalability.

Following the proposed optimization strategy in this paper, the proposed optimized Crystal-MD program used a complete potential table, so its results were more accurate than those of the original Crystal-MD program.

4 Related Work

Molecular dynamics represents a multi-body simulation method, which obtains macroscopic and microscopic process quantities, such as system structure and property, by simulating the interaction and motion process of microscopic particles in a multi-body system composed of particles by a computer. Fermi et al.[19] firstly proposed the molecular dynamics method in 1955. Verlet[20] proposed the classic Verlet algorithm in 1967, and in the molecular dynamics simulations, the acceleration, displacement, velocity, and other physical parameters of particles were calculated step by step, thus expanding and developing the research field and scope of the molecular dynamics. Daw and Baskes[21,22] proposed the famous EAM based on the density-functional theory and quasi-atomic theory. The molecular dynamics software, such as LAMMPS[2], IMD[23], Ls1-MarDyn[24], and COMD[25], has been widely used. In Ref. [11], the implementation of molecular dynamics under Intel's Xeon Phi was studied, the Single Instruction Multiple Data (SIMD) performance was improved, the OpenMP shared-memory model was implemented, and the thread cost of the OpenMP was decreased by deleting the reduce. According to the characteristics of the BCC structural-based metals, Bai et al.[4] proposed a new Lattice Neighbor List to reduce the space occupied by neighbor atomic information. The communication model of parallel MD simulation was designed for this kind of data structure. Based on this data structure and communication model, a large-scale parallel MD simulation software, the Crystal-MD was successfully developed[5], which greatly reduced the memory required for software operation. The Crystal-MD simulations achieved a parallel efficiency of more than 90% in test cases and reduced the memory of a multicore cluster by more than 25% compared to the LAMMPS and IMD, which are two popular molecular dynamics simulators.

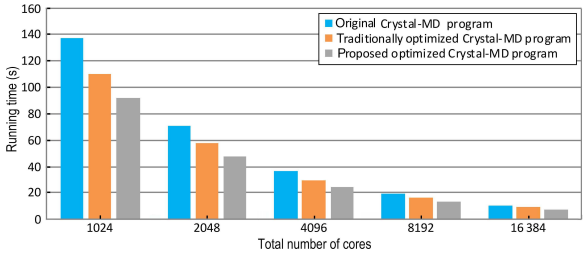


Fig. 10 Running time of the original Crystal-MD program, traditionally optimized Crystal-MD program, and proposed optimized Crystal-MD program in the medium-scale test environment.

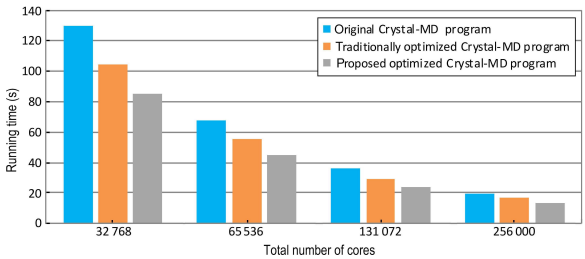


Fig. 11 Running time of the original Crystal-MD program, traditionally optimized Crystal-MD program, and proposed optimized Crystal-MD program in the large-scale test environment.

5 Conclusion

We introduce the massive parallel MD simulation software Crystal-MD and analyze the problems of the Cystal-MD on Sunway TaihuLight that exists in the process of accessing memory. In view of the problems, we divide the limited LDM into two parts: SWC and DAC. Taking the potential energy calculation between atoms as an example, we analyze the characteristics of accessing array variables and determine whether to store the data in SWC or DAC according to the characteristics. In this paper, the function of software Cache is realized by using the limited LDM in CPEs, and

DAC is optimized. Finally, compared with the original Crystal-MD program and the traditionally optimized Crystal-MD program, the memory access optimization strategy proposed in this paper can obviously improve the running efficiency and has a better optimization effect for different test scales.

Acknowledgment

This work was supported by the National Key R&D Program of China (No. 2017YFB0202003).

References

[1] B. J. Alder and T. E. Wainwright, Studies in molecular dynamics. I. General method, *J. Chem. Phys.*, vol. 31, no. 2, pp. 459–466, 1959.

[2] S. Plimpton, LAMMPS molecular dynamics simulator, http://lammps.sandia.gov/, 2004.

[3] J. Roth, F. Gähler, and H. R. Trebin, A molecular dynamics run with 5 180 116 000 particles, *Int. J. Mod. Phys. C*, vol. 11, no. 2, pp. 317–322, 2000.

[4] H. Bai, C. J. Hu, X. F. He, B. Y. Zhang, and J. Wang, Crystal MD: Molecular dynamic simulation software for metal with BCC structure, in *Proc. 1st National Conf. Big Data Technology and Applications*, Harbin, China, 2015, pp. 247–258.

[5] C. J. Hu, H. Bai, X. F. He, B. Y. Zhang, N. M. Nie, X. M. Wang, and Y. W. Ren, Crystal MD: The massively parallel molecular dynamics software for metal with bcc structure, *Comput. Phys. Commun.*, vol. 211, pp. 73–78, 2017.

[6] D. X. Chen and X. Liu, *Parallel Programming Design and Optimization of Sunway Taihulight*, (in Chinese). Wuxi, China: National Super-computing Wuxi Center, 2017.

[7] W. Q. Dong, L. T. Kang, Z. Quan, K. L. Li, K. Q. Li, Z. Y. Hao, and X. H. Xie, Implementing molecular dynamics simulation on Sunway Taihulight system, in *Proc. 2016 IEEE 18th Int. Conf. on High Performance Computing and Communications*; *IEEE 14th Int. Conf. Smart City*; *IEEE 2nd Int. Conf. Data Science and Systems*, Sydney, Australia, 2016, pp. 443–450.

[8] C. Yang, W. Xue, H. H. Fu, H. T. You, X. L. Wang, Y. L. Ao, F. F. Liu, L. Gan, P. Xu, L. N. Wang, et al., 10M-core scalable fully-implicit solver for nonhydrostatic atmospheric dynamics, in *Proc. Int. Conf. for High Performance Computing*, *Networking*, *Storage and Analysis*, Salt Lake City, UT, USA, 2016, p. 6.

[9] W. J. Hong, K. L. Li, Z. Quan, W. D. Yang, K. Q. Li, Z. Y. Hao, and X. T. Xie, PETSc's heterogeneous parallel algorithm design and performance optimization on the Sunway Taihulight system, (in Chinese), *Chin. J. Comput.*, vol. 40, no. 9, pp. 2057–2069, 2017.

[10] L. J. Jiang, C. Yang, Y. L. Ao, W. W. Yin, W. J. Ma, Q. Sun, F. F. Liu, R. F. Lin, and P. Zhang, Towards highly efficient DGEMM on the emerging SW26010 many-core processor, in *2017 46th Int. Conf. Parallel Processing*, Bristol, UK, 2017, pp. 422–431.

[11] Y. L. Ao, C. Yang, F. F. Liu, W. W. Yin, L. J. Jiang, and Q. Sun, Performance optimization of the HPCG benchmark on the Sunway Taihulight supercomputer, *ACM Trans. Arch. Code Optim.*, vol. 15, no. 1, p. 11, 2018.

[12] Y. Yu, H. An, J. S. Chen, W. H. Liang, Q. Q. Xu, and Y. Chen, Pipelining computation and optimization strategies for scaling GROMACS on the Sunway many-core processor, in *Proc. 17th Int. Conf. Algorithms and Architectures for Parallel Processing*, Helsinki, Finland, 2017, pp. 18–32.

[13] S. M. Chen, L. Peng, S. Irving, Z. Zhao, W. H. Zhang, and A. Srivastava, qSwitch: Dynamical off-chip bandwidth allocation between local and remote accesses, *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 37, no. 1, pp. 75–87, 2018.

[14] W. H. Zhang, X. F. Ji, B. Song, S. Q. Yu, H. B. Chen, T. Li, P. C. Yew, and W. Y. Zhao. VarCatcher: A framework for tackling performance variability of parallel workloads on multi-core, *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1215–1228, 2017.

[15] Y. P. Lu, X. Wang, W. H. Zhang, H. B. Chen, L. Peng, and W. Y. Zhao, Performance analysis of multimedia retrieval workloads running on multicores, *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 3323–3337, 2016.

[16] J. N. Glosli, D. F. Richards, K. J. Caspersen, R. E. Rudd, J. A. Gunnels, and F. H. Streitz, Extending stability beyond CPU millennium: A micron-scale atomistic simulation of Kelvin-Helmholtz instability, in *SC'07*: *Proc. 2007 ACM/IEEE Conf. Supercomputing*, Reno, NV, USA, 2007, pp. 1–11.

[17] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, *J. Comput. Phys.*, vol. 117, no. 1, pp. 1–19, 1995.

[18] D. Fincham, Parallel computers and molecular simulation, *Mol. Simul.*, vol. 1, nos. 1&2, pp. 1–45, 1987.

[19] E. Fermi, P. Pasta, S. Ulam, and M. Tsingou, *Studies of the NonLinear Problems. I. Los Alamos Report LA-1940*. Los Alamos, NM, USA: Los Alamos Scientific Laboratoryz, 1955.

[20] L. Verlet, Computer "experiments" on classical fluids. I. Thermodynamical properties of lennard-jones molecules, *Phys. Rev*, vol. 159, pp. 98–103, 1967.

[21] M. S. Daw and M. I. Baskes, Semiempirical, quantum mechanical calculation of hydrogen embrittlement in metals, *Phys. Rev. Lett.*, vol. 50, no. 17, pp. 1285–1288, 1983.

[22] 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.

[23] J. Stadler, R. Mikulla, and H. R. Trebin, IMD: A software package for molecular dynamics studies on parallel computers, *Int. J. Mod. Phys. C*, vol. 8, no. 5, pp. 1131–1140, 1997.

[24] Large systems 1: Molecular dynamics, https://www.ls1-mardyn.de/, 2019.

[25] Comd, http://find.nlc.cn/search/showDocDetails?docId=6785449989126485706&dataSource=crfd&query=%20Comd, 2008.

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

Panpan Du received the BS degree from Qingdao University of Technology in 2017 and the master degree from University of Science and Technology Beijing in 2020. She is currently a test developer at Dawning Information Industry Co., Ltd. Her current research interests include cloud computing and parallel computing.

Kai Zhang received the BS degree in automation from University of Science and Technology Beijing in 2016 and the master degree from University of Science and Technology Beijing in 2019. He is currently a software developer at Beijing Sogou Technology Development Co., Ltd. His current research interests include parallel computing, cloud computing, and social network applications.

Jie Lin received the BS degree from University of Science and Technology Beijing in 2018. She is currently a master degree candidate at University of Science and Technology Beijing, China. Her current research interest is parallel computing.

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.