

MD Simulation of Hundred-Billion-Metal-Atom Cascade Collision on Sunway Taihulight

MD Simulation of Hundred-Billion-Metal-Atom Cascade Collision on Sunway Taihulight

Genshen Chu^{b,a,*}, Yang Li^{b,a,**}, Runchu Zhao^b, Shuai Ren^b, Wen Yang^c,
Xinfu He^c, Chungjun Hu^a, Jue Wang^a

^a*Engineering Research Center of Intelligent Supercomputing, Ministry of Education,
Beijing, China*

^b*University of Science and Technology Beijing, Beijing, China*

^c*China Institute of Atomic Energy, Beijing, China*

^d*Computer Network Information Center, Chinese Academy of Science, Beijing, China*

Abstract

Radiation damage to the steel material of reactor pressure vessels is a major threat to the nuclear reactor safety. It is caused by the metal atom cascade collision, initialized when the atoms are struck by a high-energy neutron. The paper presents MISA-MD, a new implementation of molecular dynamics, to simulate such cascade collision with EAM potential. MISA-MD realizes (1) a hash-based data structure to efficiently store an atom and find its neighbors, and (2) several acceleration and optimization strategies based on SW26010 processor of Sunway Taihulight supercomputer, including an efficient potential table storage and interpolation method, a coloring method to avoid write conflicts, and double-buffer and data reuse strategies. The experimental results demonstrated that MISA-MD has good accuracy and scalability, and obtains a parallel efficiency of over 79% in an 655-billion-atom system. Compared with a state-of-the-art MD program LAMMPS, MISA-MD requires less memory usage and achieves better computational performance.

Keywords: computational materials, material radiation damage simulation, multi/many-core processor

PROGRAM SUMMARY/NEW VERSION PROGRAM SUM-

**E-mail address:* genshenchu@gmail.com

***Corresponding author.*

E-mail address: liyang@ustb.edu.cn

MARY

Program Title: MISA-MD

CPC Library link to program files: (to be added by Technical Editor)

Developer's repository link: (if available)

Code Ocean capsule: 10.24433/CO.4041607.v1

Licensing provisions(please choose one): BSD 3-clause

Programming language: C and C++

Supplementary material:

Journal reference of previous version: *

*Does the new version supersede the previous version?:**

*Reasons for the new version:**

*Summary of revisions:**

Nature of problem(approx. 50-250 words):

Molecular dynamics(MD) is a significant method to simulate the cascade collision progress of the key material in nuclear reactors. However, there are many difficulties for existing MD programs to perform large scale cascade collision simulations. Thus, it is especially essential to develop a new MD software to extend cascade collision simulations to larger spatial scale and longer temporal scale.

Solution method(approx. 50-250 words):

To achieve accuracy and effective MD cascade collision simulation, the EAM potential is selected to calculate interactional force between atoms in the simulation system. To extend MD simulation to larger scale, we proposed a hash-based data structure/algorithm to efficiently store an atom and find its neighbors, and several acceleration and optimization strategies based on SW26010 processor of Sunway Taihulight supercomputer.

Additional comments including restrictions and unusual features (approx. 50-250 words):

* Items marked with an asterisk are only required for new versions of programs previously published in the CPC Program Library.

1. Introduction

As a sustainable, clean and renewable energy source, nuclear energy is a promising solution to global energy crisis and environmental pollution. Nevertheless, how to ensure the safety of the nuclear reactor is a crucial issue. Many components of a reactor are exposed to a radiation environment that is of high temperature and high pressure, and is full of high-energy neutrons.

Particularly, reactor pressure vessel (RPV), a unique and non-replaceable part, is the last protection of the fission reactor core. Its integrity directly determines the service time of the whole reactor.

Radiation damage to the key material of RPV is a major threat to the integrity of RPV. Existing research revealed that the damage is initiated when a given lattice atom, namely, primary knock-on atom (PKA), is struck by a high-energy neutron [1, 2, 3, 4]. Then, PKA will continue to perform a sequence of collisions with other atoms. Afterwards, the system will generate the secondary, the third, and the subsequent higher-order knock-ons until all the energy initially imported to the PKA has been dissipated [1, 2]. This process, called *cascade collision*, usually stops within tens of picosecond ($1\text{ps} = 10^{-12}\text{s}$) approximately. It is almost impossible to observe in experiment.

To study material evolution behavior at atomic scale, molecular dynamics (MD) [5, 6] can be applied to the simulation of the *cascade collision*. Due to the high computational cost required by the simulation on large spatial scale and long temporal scale, parallel molecular dynamics simulation is especially essential. There have been a number of parallel molecular dynamics programs [7, 8, 9]. Nevertheless, existing programs are limited in the functionality (e.g. direct defect analyzing, adaptive timestep length, batch execution of simulations) and the execution efficiency. Consequently, they are not applicable to the large-scale *cascade collision* simulation.

This paper proposes MISA-MD, a new parallel MD implementation, to achieve the efficient and accurate *cascade collision* simulation on large spatial scale and long temporal scale. There are various potential functions used in MD simulation under different fields, such as Tersoff potential [10] and Lennard-Jones (L-J) potential [11], for calculating the interaction among atoms. To improve the simulation accuracy, MISA-MD adopted Embedded Atom Method (EAM) potential [12], a complex but pretty accurate potential function, which can provide an effective interatomic description for metallic system. To improve the runtime performance, MISA-MD designed and realized a new hash based data structure for efficient atom storage and quick neighbor atom indexing. Several acceleration and optimization strategies were also applied to ensure that MISA-MD can make full use of SW26010 processors on Sunway Taihulight supercomputer.

The key contributions of this paper are summarized as follows:

1. A new hash based data structure for storing and indexing lattice-based

atoms;

2. A new method of EAM potential tables storage and interpolation on SW26010 processor;
3. A new coloring method to avoid the write conflicts on computing processing elements (CPEs) of SW26010 processor.
4. New double-buffer DMA and data reuse strategies to reduce the overhead of data transmission from main memory to local device memory (LDM) of CPEs.

The remainder of this paper is organized as follows. Section 2 presents the background knowledge on MD and Sunway Taihulight supercomputer, and discusses the related work. Section 3 introduces the new atom storage and neighbors indexing data structure. Section 4 discusses the implementation on Sunway TaihuLight supercomputer and the optimization strategies. Section 5 presents a performance analysis and validation of MISA-MD. The last section summarizes the conclusion and the future work.

2. Background

2.1. Molecular Dynamics Method and Challenges

Molecular dynamics is a classic method for simulating particle systems. It has been widely used in many domains, such as material science, chemistry, and biomolecular science [13]. A generic MD workflow is shown in Fig. 1. In MD, an atom/molecule is treated as a particle. For initialization, each particle is created with an initial coordinate and a velocity. Then, the computation falls into a loop of time steps to solve Newton’s equations of motion. In each time step, the computation updates every particle as follows. First, calculate the force applied to each particle to solve its acceleration a . The force can be calculated by interactional potential functions (e.g. Lennard-Jones potential and EAM potential) in a particular system. We will discuss the potential function in Section 2.2 in details. Second, update the particle velocity via the integral of acceleration in a short Δt , where Δt is the length of a time step. Third, update the particle coordinate using the integral of velocity in Δt . Finally, the corresponding ensemble, such as NVT or NPT ensemble, is applied, and some physical quantities may also be calculated before the loop moves into the next time step.

The interaction forces to be computed among particles can typically fall into two categories: long-range and short-range [14]. Long-range force calculation requires interactions of each particle contributed by the global system.

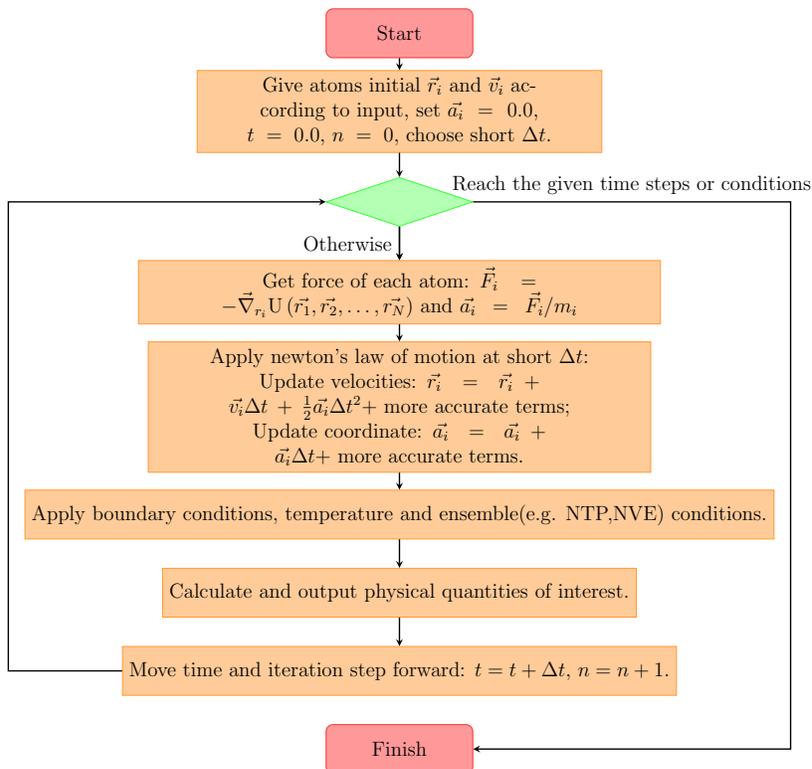


Figure 1: A standard workflow of MD applications. In the time step loop, it will repeat to calculate force, update velocities and position and perform constraint condition as they need.

While in short-range force calculation, for any particle p , only its neighbor particles that are located within a cut-off radius is considered to contribute to the force that applied to p . In fact, short-range force calculation is adopted by most particle systems in MD simulation, including particle interaction modeled by EAM or L-J potential.

Although most MD implementations are parallel, it is still challenging to realize the simulation of large spatial scale and long temporal scale. For spatial scale, the simulation is mainly restricted by memory storage. Given a fixed memory capacity, the more particles can be stored (i.e, the less memory is required by a single particle), the larger spatial scale MD simulation can reach. For temporal scale, the faster the computation of each time step is, the more time steps can be computed within the same time limitation. For MD simulation with computation of short-range interaction forces, the time

cost of each time step is mainly affected by neighbor particle indexing and the force calculation. To reduce the time cost of each time step, we must manage the particle data and index neighbor particles as efficiently as possible.

2.2. EAM Potential

Different from pair potentials (e.g. Lennard-Jones and Morse), embedded-atom method potential is able to provide a better interatomic description for metallic system. In EAM, the energy of an atom i is determined by two aspects [12]: (1) the embedded energy of the atom embed in electron cloud which represents the *many-body effects* in the interaction, and (2) the pair-wise potential of atomic interaction. EAM potential can be more complex compared with pair potentials. In Eq. (1), E_i denotes the energy of atom i , and r_{ij} represents the distance between atom i and its neighbor atom j . Three kinds of functions are presented in Eq. (1):

1. $\rho_\beta(r)$: It describes the contribution to the electronic density at the site with distance of r from atom j whose type is β .
2. $F_\alpha(\rho)$: The embedding energy function F returns the energy associated with placing an atom of type α in the electron environment described by ρ .
3. $\phi_{\alpha\beta}(r_{ij})$: $\phi_{\alpha\beta}$ is pair-wise potential function, which describes the pair potential between two atoms i of type α and j of type β . And r_{ij} denotes the distance of atom i and j .

$$E_i = F_\alpha \left(\sum_{j \neq i} \rho_\beta(r_{ij}) \right) + \frac{1}{2} \sum_{i \neq j} \phi_{\alpha\beta}(r_{ij}) \quad (1)$$

The total force \vec{F}_i applied to atom i can be expressed by the negative gradient of potential energy of the simulation system, as described in [15, 16]. Then, we can calculate the force that is applied to atom i from neighbor atom j using Eq. (2), where: $F_i(\rho)$ and $\rho_i(r)$ denote the embedding energy function and electronic density function associated with type of atom i respectively; $\phi_{ij}(r)$ denotes the pairwise potential function associated with types of atom i and j ; \vec{r}_i and \vec{r}_j represent the position vector of atom i and j , respectively, and r_{ij} is the distance of atom i and j ; $\bar{\rho}_i$ denotes the electronic density contributed by all neighbor atoms of atom i .

In a system with n atom types, $n \times (n + 1) / 2 + n + n + n$ functions should be determined, including $n \times (n + 1) / 2$ partial derivative of pairwise potential

$$\vec{F}_{ij} = - \left[\frac{\partial F_i(\rho)}{\partial \rho} \Big|_{\rho=\bar{\rho}_i} \frac{\partial \rho_j(r)}{\partial r} \Big|_{r=r_{ij}} + \frac{\partial F_j(\rho)}{\partial \rho} \Big|_{\rho=\bar{\rho}_j} \frac{\partial \rho_i(r)}{\partial r} \Big|_{r=r_{ij}} + \frac{\partial \phi_{ij}(r)}{\partial r} \Big|_{r=r_{ij}} \right] \frac{\vec{r}_i - \vec{r}_j}{r_{ij}} \quad (2)$$

functions, n partial derivative of embedding energy functions, n electronic density functions and n partial derivative of electronic density functions. For example, a system with two different atom types (i.e., α and β), we have to calculate following 9 functions: $\rho_\alpha(r)$, $\rho_\beta(r)$, $\frac{\partial \phi_{\alpha\alpha}(r)}{\partial r}$, $\frac{\partial \phi_{\alpha\beta}(r)}{\partial r}$, $\frac{\partial \phi_{\beta\beta}(r)}{\partial r}$, $\frac{\partial F_\alpha(\rho)}{\partial \rho}$, $\frac{\partial F_\beta(\rho)}{\partial \rho}$, $\frac{\partial \rho_\alpha(r)}{\partial r}$, $\frac{\partial \rho_\beta(r)}{\partial r}$.

To calculate the force of each atom using EAM potential according to Eq. (2), the following steps must be performed:

1. Calculate electronic density contribution from neighbor atoms for each atom i : $\bar{\rho}_i = \sum_{k \neq i} \rho_k(r_{ik})$;
2. Calculate value of partial derivative of embedding energy function for each atom i at $\bar{\rho}_i$: $\frac{\partial F_i(\rho)}{\partial \rho} \Big|_{\rho=\bar{\rho}_i}$;
3. Calculate value of partial derivative of pairwise potential function for every two atoms i and j in cut-off radius: $\frac{\partial \phi_{ij}(r)}{\partial r} \Big|_{r=r_{ij}}$;
4. For every two atoms i and j in cut-off radius, calculate each other's contribution value of partial derivative of electronic density function via: $\frac{\partial \rho_i(r)}{\partial r} \Big|_{r=r_{ij}}$ and $\frac{\partial \rho_j(r)}{\partial r} \Big|_{r=r_{ij}}$;
5. Here, all terms in Eq. (2) are available, then we can calculate force \vec{F}_{ij} of two atoms i and j in cut-off radius via Eq. (2). For each atom i , traverse all its neighbor atoms j , add force contributed by j to atom i . Then we can obtain the total force for each atom.

2.3. Sunway TaihuLight and SW26010 Many-Core Processor

Sunway TaihuLight [17, 18] is a supercomputer with a peak performance of 125.3 PFlops and a sustained Linpack performance of 93 PFlops. It was manufactured by *National Research Center of Parallel Computer Engineering and Technology (NRCPC)*, and is hosted at National Supercomputing Center

in Wuxi. It comprises 40960 homegrown SW26010 many-core processors [18] which are connected by a 2-level fat-tree topology network.

The SW26010 processor uses on-chip heterogeneous many-core architecture. A SW26010 consists of four core groups (CGs) that are connected via the network on chip (NoC) and a system interface (SI) [19]. A CG consists of a management processing element (MPE), a computing processing element (CPE) cluster, and a memory controller (MC). Every CG has its own memory space, and the main memory is connected to the MPE and CPE cluster through the MC. Each CPE cluster contains 64 CPEs that are arranged in an 8×8 grid. The SI is used to connect the processor itself and other devices outside.

The MPE and CPE are both 64-bit reduced instruction set computer (RISC) cores. CPE is simpler than MPE and can only run in user mode. Moreover, CPE is a core with only 64 KiB user-controlled local device memory (LDM), while MPE is connected with a 8 GB DDR3 memory via MC. For memory accessing, CPE can access main memory directly by global load/store instructions (gld/gst), or by direct memory access (DMA) which can be much faster than gls/gst. At CPE cluster level, each CPE has 8 column communication buses and 8 row communication buses to provide fast register communication across the 8×8 CPE grid, which is a significant capability to share data inside CPE cluster.

2.4. Related Work

Many molecular dynamics programs were developed by numerous research teams. Sandia National Labs developed LAMMPS [7] for classical molecular dynamics simulation. However, the overhead of maintaining the neighbor list is very high during a large-scale simulation (e.g. 10^{10} particles). Duan et al.[8] adapted LAMMPS for Sunway many-core architecture. The new program achieved over 2.43 PFlops performance for a Tersoff simulation using 16,384 nodes. However it only supports L-J potential and Tersoff potential, rather than the EAM potential that is more suitable for metallic systems. GROMACS[20] is a widely-used molecular dynamic program in chemical and biomolecular. SW_GROMACS[21] accelerated GROMACS on Sunway TaihuLight supercomputer. Besides GROMACS, another significant MD software package for biomolecular simulation is NAMD [22], it is written using the Charm++ parallel programming model [23] and is the recipient of Gordon Bell award in 2002. ls1 Mardyn[9] is another massively parallel molecular dynamics program using L-J potential[11] to solve the interaction

among particles. Its main target is the simulation of thermodynamics and nanofluidics. It achieved a 2.1×10^{13} -particle simulation on Hazel Hen supercomputer by using 172,032 CPU cores[24]. However, both GROMACS and ls1 Mardyn cannot simulate the metallic systems with EAM potential. Crystal MD[25, 26] is a parallel MD program for metal with BCC structure. It carried out a four-trillion-atom simulation on Sunway TaihuLight supercomputer. SPaSM [27] is a MD implementations and can achieve simulation of more than 100 billions on BlueGeneL. But there is a lack of functionality of cascade collision simulation for both CrystalMD and SPaSM.

To support indexing of neighbor atoms within a cut-off radius, several classical data structures were proposed, such as the neighbor list (or verlet lists)[28] used in LAMMPS , NAMD and GROMACS, and link cell [29, 9] used in ls1 Mardyn and SPaSM. In the neighbor list, each atom will create and maintain a reference to its neighbor atoms and store them in a list. The neighbor list requires a lot of memory. Link cell splits the simulation box into a number of cells. In force computation step, for any atom a in cell c , link cell must traverse all atoms in cell c and all atoms in neighbor cells of c . which may index many unnecessary atoms out of cut-off radius. Thus new design of data structure for efficient atoms storing and quick neighbor atoms indexing is essential.

3. Hash Based Data Structure

To support efficient atom storing and quick indexing of neighbor atoms, a new hash-based data structure is designed for MISA-MD. The hash-based data structure consists of two parts. The first one is Lattice List used to efficiently store atoms, and the second one is Neighbor Offset Index targeting quick neighbor atom indexing. In lattice list, a lattice array will be established as a hash array, and a red-black tree will also be setup for storing elements with hash collision. The hash function maps each atom to its nearest lattice in cartesian coordinate system. Fig. 2 illustrates this idea in a 2d simulation box. For neighbor offset index, the offset value of lattice id is pre-calculated for later neighbor atoms indexing. To traverse neighbor atoms of a atom, the id of nearest lattice of the atom will be calculated first, then we can obtain neighbor lattices by adding the nearest lattice id and offset lattice id. Fig. 3 shows the lattice offset for calculating neighbor atoms.

The complete process of MD calculation based on BCC structure using lattice list and neighbor offset index data structure is list as following.

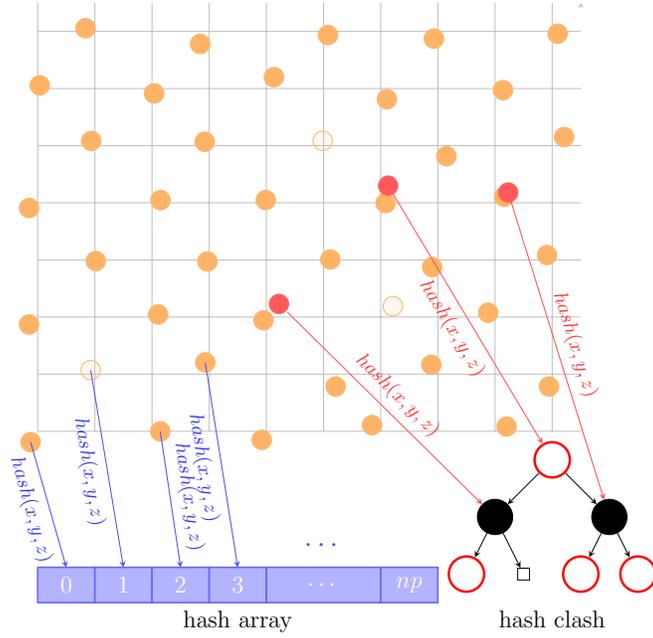


Figure 2: Atoms data structure and indexing method

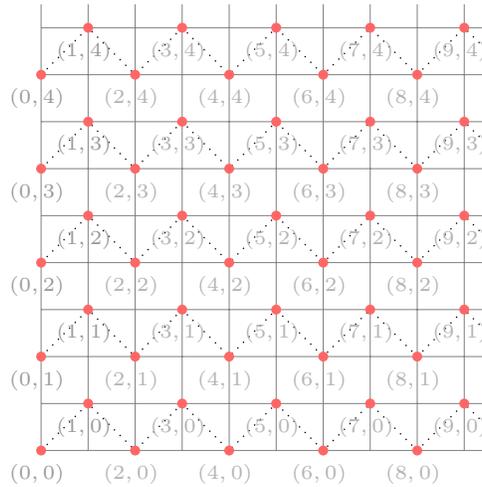


Figure 3: Lattice Offset Index for calculating neighbor atoms for BCC structure in 2d domain. The number in brackets under each lattice site is the coordinate of the corresponding lattice site

Hash Initialization. At the beginning, a hash array for lattice will be allocated. In our cascade collision simulation, body centered cubic (BCC) structure lattice is constructed for iron-based or tungsten-based material, in which there is an additional lattice site located in the center of each cube unit. Each element in hash array corresponds to a BCC lattice site. For example, in the 2d simulation domain¹, the *coordinate* of each lattice site can be recorded, as shown in Fig. 3, and the coordinate can then be converted to hash array index by:

$$index = 2 \times box_x \times y + x$$

where box_x and box_y are the lattice size of simulation box of current MPI process at x and y dimension, respectively. and x and y are the corresponding lattice coordinates at each dimension. In Fig. 3, box_x and box_y are both equal to 5, a lattice with coordinate (3, 4) is mapped to the 43rd ($2 \times 4 \times 5 + 3 = 43$) element in hash array. And in 3d simulation domain, the index can be calculated by:

$$index = 2 \times box_x \times box_y \times z + 2 \times box_x \times y + x \quad (3)$$

The advantage of this BCC lattice to index mapping method is that we can store all lattice associated atoms in a compact memory mode. We call the calculated index of hash array as *lattice id*. Meanwhile, the lattice id can also be converted to lattice coordinate easily.

The hash function can calculate lattice coordinate of the nearest lattice site of a atom and return lattice id of the nearest lattice site. Hash clash occurs when two distinct atom coordinates passed into the hash function produce identical lattice id outputs. A empty red-black tree will be created for storing hash clash atoms and indexed by lattice id. In this case, one of these atoms is stored in hash array and the left ones are stored in the red-black tree when hash clash occurs.

Atoms are organized by this hash table. In system initialization step of simulation, all atoms are placed at the corresponding lattice site. In other words, there is no hash clash. It is because that any two atoms must correspond to different lattice sites. Therefore, all atoms data are stored in hash array at initialization step.

¹In fact, it is a 3d domain in practice.

Neighbor Offset Index Initialization. Under this regular lattice structure, list of offset lattice ids for indexing neighbor lattices in cut-off radius can be pre-calculated.

Iterate Neighbor Atoms. We can calculate force or other interactions of each atoms in simulation domain via lattice list and neighbor offset index. Algorithm 1 show the algorithm of iterating all atoms and neighbor atoms to calculate force.

Note that, for obtaining interaction contributions of neighbor atoms from neighbor MPI processes conveniently, ghost region is involved. The atoms in ghost region is also indexed by lattice list together with local atoms located in real simulation domain.

In the algorithm, the calculation consists of three parts: interactions of atoms from both hash array, interactions of atoms from hash array and hash clash, and interactions of atoms from both hash clash. In the first part of interactions computation, just iterate over the hash array and traverse neighbor lattices of each lattice to obtain interaction contribution from neighbor atoms that are stored in hash array. In interactions calculation of atoms from hash array and hash clash, iterate atoms in hash clash, if the atom is not in ghost area, find its nearest lattice via hash function, then we can traverse neighbor atoms in hash array by the nearest lattice as center lattice. At last, it falls into the interactions of atoms in hash clash. It iterate the atoms that are not in ghost region. For each atom a in iteration, we can find the nearest lattice of this atom via hash function. Then we can calculate ids of neighbor lattices by id of the nearest lattice and precalculated offset lattice ids, and select atoms in hash clash with the neighbor lattice ids, and add interactions contribution of these ones to atom a .

Hash Update. After finishing each simulation step, the atoms coordinate can get changed, thus updating of hash index is necessary. Algorithm 2 presents the steps of hash updating. The hash updating can be divided into two steps: 1) Traverse all elements (or atoms) in hash array. For each valid element, re-calculate its lattice id via hash function. If array index of the current element in hash array does not equal to the new calculated lattice id, just copy this element to hash clash and tag this element as invalid. 2) Traverse all of atoms in hash clash. For each atom in hash clash that is not located in ghost region, we can calculate its lattice id via hash function. If the element in hash array indexed by the re-calculated lattice id is invalid, then copy the

atom data to hash array, set the element in hash array as valid, and remove this atom from hash clash.

These two steps above in Algorithm 2 can make sure: 1) If two or more than two atoms have the identical hash value, only one of them is stored in hash array, and the left ones are stored to hash clash. 2) If there is a lattice id (index value in hash array) but no atom in simulation domain corresponds to this lattice id, the element indexed by the lattice id in hash array must be invalid. 3) For each non-ghost atom in hash clash, if there is a invalid element in hash array that is indexed by the hash value of this atom, the algorithm can remove the atom from hash clash and add it back to hash array.

The advantage of the hashed-based approach is that it can support quick neighbor searching and efficient atom storing, which use less memory compared than neighbor list method. Denote M as the number of neighbor in cut-off radius and N as the number of atoms in simulation system. The memory occupation can be divided as two parts: system atoms and neighbor indexing. For system atoms, both the new hash-based approach and the neighbor list method need $O(N)$ memory to save system atoms. But for neighbor indexing, the memory cost of the new hash-based approach will be $O(M)$, and memory cost of the neighbor list method will be $O(MN)$. However, the disadvantage is also obvious. The hash-based approach is specific to solids and would not work well for the system where the atoms are not on a regular lattice, such as liquids. Because it will cost expensive computation to handle hash claims when the atoms are not on a regular lattice.

4. Sunway Acceleration and Optimizations

To achieve efficient MD simulation on large-scale clusters and make full use of the computation ability of SW26010 processor, accelerating computation of MISA-MD is essential. Results from profile tools show that the EAM potential computation in MISA-MD, consume more than 80% computation time. In SW26010 processor, 256 CPEs in 4 core group contribute more than 90% peak performance of the overall processor[30]. Therefore, the code for accelerating EAM interaction computation on sunway CPEs is implemented. In current version of MISA-MD, we only consider the acceleration for interaction of atoms from hash array due to the fact of low rate of hash clash.

Moreover, it is of great significance to accelerate EAM computation on SW26010. Because it is another significant approach to extend MD simulation to longer temporal scale besides quick neighbor atoms indexing.

4.1. Potential Tables Storage and Interpolation

In EAM computation, the $n \times (n + 1) / 2 + n + n + n$ functions for n different atom types are determined by spline interpolation from a table with dispersed values called “potential table”.

For each to be determined function as well as its partial derivative, N equidistant dispersed points are presented in potential table. For instance, to determine electronic density function of atom type β , point set $P_\beta = \{(r_1, \rho_1), (r_2, \rho_2), (r_3, \rho_3), \dots, (r_N, \rho_N)\}$ is given in table, where r_i is distance and ρ_i is value of electronic density under distance r_i , $i = 1, 2, \dots, N$. We are expected to find a electronic density function $\rho_\beta(r)$ and its partial derivative function $\frac{\partial \rho_\beta(r)}{\partial r}$ for $r \in [r_1, r_N]$.

To perform interpolation on a point set $P = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_n, y_n)\}$ with condition of $x_1 < x_2 < \dots < x_N$ and $x_2 - x_1 = x_3 - x_2 = \dots = x_N - x_{N-1} = h$, a cubic polynomial function $S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$ is selected for each segment specified by two adjacent point to approximate the undetermined function in spline interpolation. Then we can calculate the coefficients a_i , b_i , c_i , and d_i of $S_i(x)$ on each segment, as well as coefficients e_i , f_i and g_i for partial derivative of $S_i(x)$: $S'_i(x) = e_i(x - x_i)^2 + f_i(x - x_i) + g_i$. After each functions is determined, then these seven coefficients on each segment can be stored for later EAM computation.

In our cases, to simulate cascade collision of a pure Fe system using MISA-MD, a potential table file from [31] is provided, which contains 3 potential tables (one pair-wise potential table for Fe-Fe, one for embedding energy for Fe and one for electronic density for Fe). Each potential table above contains 5000 dispersed data values used for interpolation to determine corresponding potential function and it partial derivative function. This means that, we need at least 820 KiB memory to store the interpolation coefficients for single-atom-type system (3 potential tables \times 7 coefficients \times 5000 segments \times 8 bytes per double precision floating point value), which is much large than 64 KiB LDM of CPE. Further, if the target system is an alloy system containing multiple types of atoms, the storage of coefficients can be much larger. Thus, the interpolation coefficients can not be loaded into the LDM of CPE at one time.

To overcome the small LDM of CPE, we copy the origin values of one table to LDM of CPE before using it, whose size is only 39 KiB. This thought is inspired by [32]. Then, all the coefficients of $S_i(x)$ and $S'_i(x)$ can be cal-

culated on the fly using the origin data and interpolation method. But the shortcoming of this method is obvious. It can only calculate single-atom-type system, because for alloy system, the LDM of CPE cannot hold all necessary origin data.

To deeply optimize the memory storage of origin potential table data, we exchange order of offset list iteration and atoms iteration in Algorithm 1 (move the offset list iteration to the outermost loop and move atoms iteration of x,y,z dimension to inner loop). Because the origin potential table is associated with atoms distance r , in each offset list iteration, we can obtain the minimum and maximum distance of neighbor atoms in atoms iteration firstly, and then we can only load part of the origin potential data that is in this distance range, rather than the whole origin data. With partial potential data loading in each offset list iteration, we can free up more memory space that can be used for atoms storage. But the number of potential data copies from main memory to LDM can increase due to the partial potential data loading.

To overcome the large overlap of potential data copies, we divide the origin data of potential tables into some blocks and each CPE pre-load a block of origin potential tables at initialization step. Then the origin potential tables are stored on CPEs distributedly. And in latter offset list iteration, the partial potential data is loaded from some specific CPEs via fast register communication, instead of main memory.

4.2. Tasks Assignment on CPEs

In fact, Newton’s third law can be applied for force calculation. As described at the end of Section 2.2, thanks to Newton’s third law, the amount of calculation of atoms’ electronic density contribution at step 1) and force contribution at step 5) as well as pair-wise potential at step 3) and derivative of electronic density at step 4) can halve. But the problem of threads’ write conflict on sunway CPEs must be solved before involving this trick. When we assign computation tasks of one MPI processor to 64 CPE threads (In sunway native programming, one CPE can launch a light-weight working thread using Athread library [33].), spatial decomposition of simulation domain is achieved. We may split the domain owned by the MPI processor into 64 blocks, and each CPE thread would carry out the EAM calculation of one block of them. But the write conflict will occur when writing calculated results back to main memory on MPE if there is common neighbor atom for two adjacent CPE threads, as shown in Fig. 4.

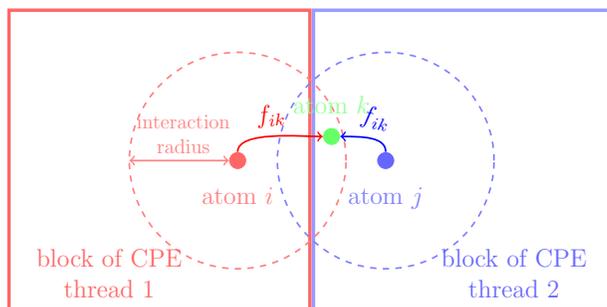


Figure 4: Write Conflict of CPE threads

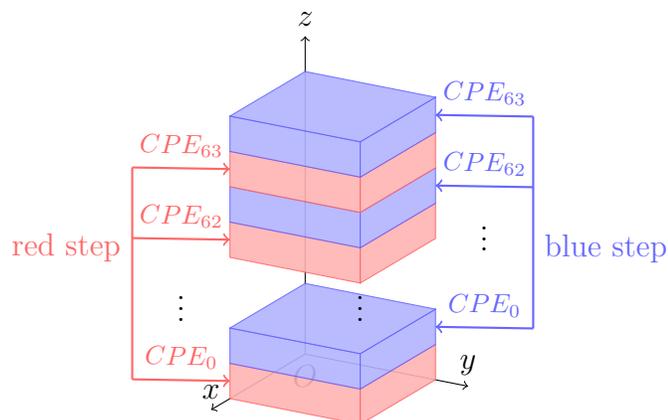


Figure 5: Coloring CPE threads

To overcome this difficulty, we proposed a coloring method for sunway CPE threads, which is inspired by SDC (Spatial Decomposition Colouring) method[34, 35] used in OpenMP program model on multi-core platforms. Fig. 5 shows an example of the coloring result. In coloring method, we split the domain space into 128 blocks in z dimension, where 128 is twice of the number of CPEs, and each block is colored as red or blue and make sure there is no adjacent blocks with the same color. Then the calculation is divided into two steps to avoid write conflict. In the first step, each CPE thread is assigned a block with red color, and the next step, each CPE thread is assigned a block with blue color. With these two steps, all blocks can be calculated without write conflict.

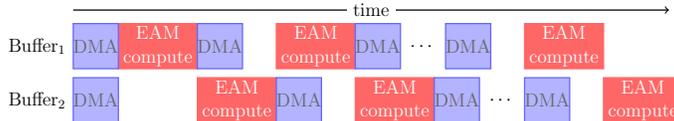
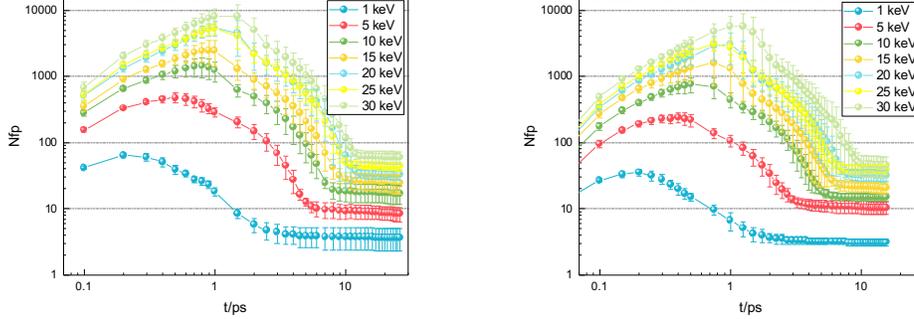


Figure 6: Double-buffer DMA strategy

4.3. Double-Buffer DMA and Data Reuse Strategies

Naturally, the domain block assigned to a CPE may contain a large number of atoms which can not be transmitted to CPE at one time due to the restriction of 64 KiB LDM. Thus, the domain block is split into numbers of small data block for CPE computation. In traditional method, CPE would request a memory block of atoms data from main memory, then it start computation once the atoms data transmission finish. After computation completes, atoms data will put back to main memory and then prepare to access next data block for next round computation repeatedly until all data blocks assigned to this CPE is completed. We found that the repeatedly DMA data accessing has a high overhead, which may leading a poor performance. Therefore, a double-buffer DMA strategy is taken to overlap the DMA data transmission and EAM potential computation, as shown in Fig. 6. In double-buffer DMA strategy, the LDM is divided into three blocks, one is for origin potential data, the second and the third ones are both buffer memory for each other. Except for the initial DMA getting and the last DMA putting operation, when performing computation of current round, the DMA putting operation of previous round or the DMA getting operation of next round can be performed simultaneously. By overlaping DMA data transmission and EAM potential computation, the computation efficiency on CPEs is greatly improved.

Double buffer strategy can overlap DMA and computation effectively. To reduce the overhead of DMA further, another optimization, called data reuse, is applied. Considering atoms data block of current calculation round, a part of them can be actually the ghost atoms data of next calculation round, and its ghost data can also be the ‘real’ atom data of next round. Thus, we can keep the these two part of data in the LDM of CPE for usage of next round calculation, which can reduces the amount of data transferred to LDM in each DMA operation.



(a) Evolution of number of displacement cascades' defects for different PKA energy from MISA-MD (b) Evolution of number of displacement cascades' defects for different PKA energy from LAMMPS

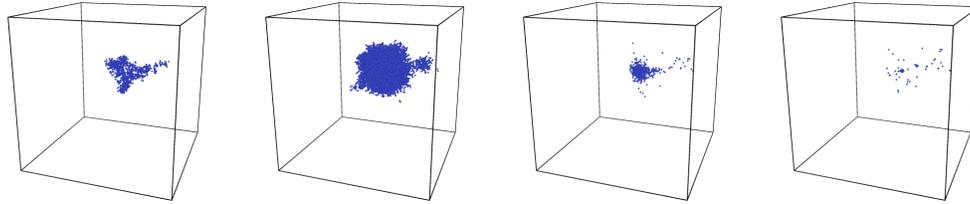
Figure 7: Evolution of number of displacement cascades' defects for different PKA energy at 600K temperatures

5. Evaluation

This section presents the evaluation on single core group performance, scalability, and accuracy of cascade collision simulations. As a comparison, we also compared memory usage and performance with LAMMPS on Sunway and Intel platforms.

5.1. Accuracy and Case Study

In order to validate the accuracy of MISA-MD, we have performed several cascade collision simulations of different PKA energies and directions. In our cases, the PKA energy settings are 1.0 keV, 5.0 keV, 10.0 keV, 15.0 keV, 20.0 keV, 25.0 keV and 30.0 keV, and PKA directions includes $\langle 1, 2, 2 \rangle$, $\langle 1, 3, 5 \rangle$ and $\langle 2, 3, 5 \rangle$. In each simulation instance, a $80a_0 \times 80a_0 \times 80a_0$ (where a_0 is lattice constant) simulation box containing 1 024 000 Fe atoms is involved, and the primary knock-on atom is placed at position of $(40a_0, 40a_0, 40a_0)$ (center of simulation box) with a corresponding velocity associated with PKA energy. For each PKA energy and each PKA, simulations are performed three times to obtain the average results and analyze the number of frenkel pairs in simulating box over evolution time. As shown in Fig. 7a. Besides, we also tested on LAMMPS using EAM potential with the same input parameters. Fig. 7b shows the evolution of frenkel pairs numbers for different PKA energies at 600K temperatures from simulation results of LAMMPS.



(a) system configuration after 0.1 ps of PKA generation within 424 Frenkel pairs. (b) system configuration after 0.8 ps of PKA generation within 3307 Frenkel pairs. (c) system configuration after 5.5 ps of PKA generation within 190 Frenkel pairs. (d) system configuration after 26.0 ps of PKA generation within 26 Frenkel pairs.

Figure 8: MISA-MD visualization of system evolution at different time under PKA energy of 15keV and direction of $\langle 1, 3, 5 \rangle$.

Comparing Fig. 7a and Fig. 7b, their evolution curves of frenkel pairs number analyzed from simulation results have the same trend under varies of PKA energies. The number of frenkel pairs of MISA-MD and LAMMPS both grow to a peak rapidly after a short time of PKA generation and then quench from peak value to stable value. As a case of this process, Fig. 8 visualizes process of cascade collision simulated by MISA-MD, in which the spatial distributions of frenkel pairs at different evolution stages under PKA energy of 15keV and incidence direction of $\langle 1, 3, 5 \rangle$ are presented. Moreover, with a large PKA energy, the peak would appear later with a larger peak, and a higher value of frenkel pairs survives at stable stage. By comparing peak time, peak value, stable time and stable value of MISA-MD and LAMMPS quantitatively, they are all similar and within acceptable tolerances. And the number of survived frenkel pairs pairs also meet the classical NRT model established by Norget [36].

5.2. Single Core Group Evaluation

To test the computational efficiency of code accelerated by 8×8 CPEs, we performed performance test on single core group with and without CPEs acceleration. In this test case, a $200a_0 \times 200a_0 \times 256a_0$ (where a_0 is lattice constant) simulation box was involved and simulations of 10 time steps was performed. Table 1 shows the execution time and speedup of case with MPE only and case with MPE plus 64 CPEs, which indicates our MISA-MD code obtains approximate 56 acceleration speedup with computation on CPEs.

5.3. Comparison with LAMMPS

In this subsection, we compare the memory usage and computational performance of MISA-MD and LAMMPS on Sunway and Intel platforms. For LAMMPS, the current latest version, *stable_3Mar2020*, is selected to be compared. As the EAM computation of LAMMPS does not supported Sunway CPEs architecture, thus, we perform performance tests of LAMMPS and MISA-MD using MPEs only on Sunway platform.

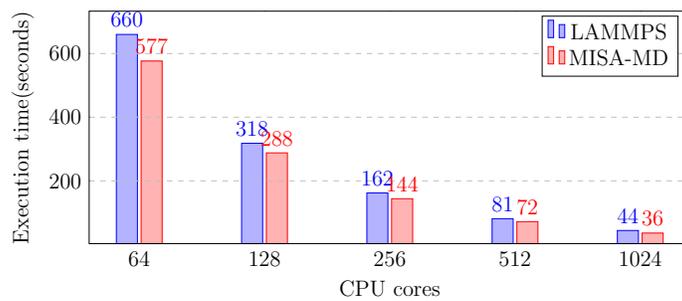
5.3.1. Memory usage

The memory usage of MISA-MD and LAMMPS is compared at runtime to demonstrate the superiority of our hash based data structure. We first apply the same scale of BCC box to MISA-MD and LAMMPS, and monitor the average memory usage of both programs. In case 1 and case 2 in Table 2, simulation box of $600 \times 600 \times 600$ containing 4.32×10^8 atoms is applied, and 120 MPI processors is involved for both MISA-MD and LAMMPS. The results of case 1 and case 2 show that the memory usage of LAMMPS is 2.7 times larger than MISA-MD. In the second test case set, as shown in case 3 and case 4 in Table 2, we use the full node memory for both MISA-MD and LAMMPS simulation, and record the maximum simulation box they can reach. The results show that, given the same memory resource, MISA-MD can simulate much larger scale than LAMMPS. In more detail, the number of atoms MISA-MD can simulate is 4.39 times than LAMMPS can reach.

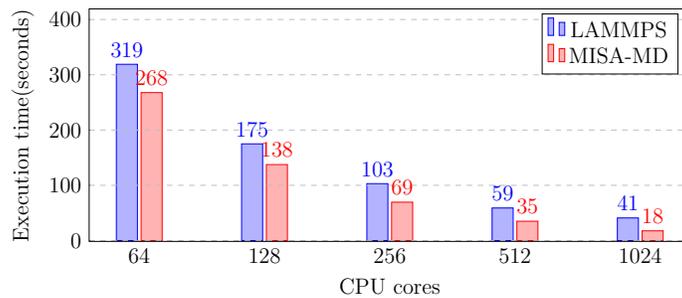
For computing efficiency, LAMMPS uses neighbor lists (Verlet lists) to keep track of nearby particles. In the above test cases, the average number of each atom's neighbor is 56 in LAMMPS. Thus, it would takes $56 \times 4 = 224$ bytes to index neighbor atoms for each atom, in which each neighbor index takes 4 bytes. In test case 4 in Table 2, the neighbor index memory occupies 38.86% of the total memory. While in MISA-MD, each atom only occupy 104 bytes for storing basic atoms information (such as velocity, electron charge density), and neighbor atoms is indexed by index offsets which only consume several KiB memory. Therefore, the new hash based data structure in MISA-MD can efficiently reduce the memory usage and can expand MD simulation to larger scale.

5.3.2. Computational Performance

The execution time is another impact to expand the scale of MD simulation. We expect simulation program can execute as fast as it could, thus it can simulate more time steps in the same spatial scale or finish as soon as



(a) Computational performance of MISA-MD and LAMMPS on Tianhe 2.



(b) Computational performance of MISA-MD and LAMMPS on Sunway Taihulight. Acceleration of MISA-MD using CPEs is disabled for principle of fairness.

Figure 9: Computational performance comparison of MISA-MD and LAMMPS on Sunway Taihulight and Intel platforms with x axis logarithmic scaled. Simulation time values are annotated on the top of bars.

possible in the same temporal and spatial scale under the given computing resources.

To compare the performance between MISA-MD and LAMMPS, test cases with fixed number of atoms are designed for both MISA-MD and LAMMPS simulation. We start our tests for MISA-MD and LAMMPS on Sunway-Taihulight and TianHe-2 supercomputer. On Sunway-TaihuLight, both MISA-MD and LAMMPS were compiled using “-O2” optimization flag by sw5gcc/sw5g++(a customized C/C++ compiler based on gcc version 5.3.0). It is worth noting that, CPEs acceleration of MISA-MD is disabled for principle of fairness because of the lack of EAM calculation acceleration on CPEs of LAMMPS. On TianHe-2, both the programs were also compiled using “-O2” optimization flag by the same compiler(gcc 5.3.0).

By giving varies of atoms number and involving 64, 128, 256, 512 and 1024 CPU cores respectively for both MISA-MD and LAMMPS, and recording execution time of both programs, the results of computational performance are obtained. The test cases on Tianhe-2 supercomputer use a $512 \times 512 \times 512$ box containing 2.68×10^8 atoms for varies of CPU cores. While for test cases on Sunway-Taihulight, simulation box of $256 \times 256 \times 256$ containing 3.36×10^7 atoms is introduced.

The performance results for both MISA-MD code and LAMMPS code on Intel and sunway platform are shown in Fig. 9. On Sunway platform, compared with LAMMPS package, computational performance of MISA-MD is increased by 19.02% to 132.56% on Sunway-Taihulight. On Intel platform of TianHe 2, the performance increment of MISA-MD is from 10.46% to 22.67% compared with LAMMPS packages. Results indicate that, **MISA-MD has a better performance than LAMMPS for all performance test cases above.**

In LAMMPS, the neighbor list is updated every few timesteps and its cost can be quite expensive. While in MISA-MD, hash index can be updated in each timestep, but the time consuming of hash index updating is much less than neighbor list updating in LAMMPS if the system does not change very frequently. In cascade collision simulation of metal materials, position of most atoms does not change very frequently. Thus, in hash index updating stage, most atoms’ hash index does not need to be updated, which can lead a better performance of MISA-MD than LAMMPS. We should also point out that simulation of other types of crystal material can also be realistic and operable, and can achieve a quite respectable performance.

5.4. Scalability

In this subsection, we will discuss our study on the scalability of our MISA-MD code on Sunway TaihuLight supercomputer. We use cascade collision cases of pure α -Fe system at temperature of 600 K as the case for the scalability tests.

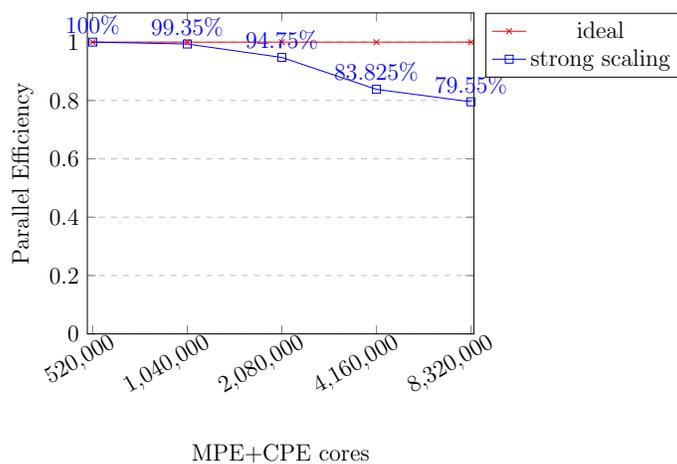
For the strong scalability, a simulation box with 655 billion (6.55×10^{11}) atoms is presented and take the performance of 520 000 cores (8000 core groups) as baseline. Fig. 10a shows the parallel efficiency of this case for strong scalability when the number of cores(including CPE cores) varies from 520 000 to 8 320 000. We can found that when the number of atoms per core group drops to 1/16 of the number, the parallel efficiency drops from 100% to 79.55%, which is quite great for the parallel efficiency.

The weak scalability performance is shown in Fig. 10b. In the weak scalability tests, we initialize the simulation system with 8.39×10^6 atoms per core group (MPI process) and take the performance of 1040 cores (16 core groups) as baseline. It can be seen that there is almost no performance loss as the scale increase. The parallel efficiency only goes down to 98.97% when the number of cores is increased to 512 times of the baseline.

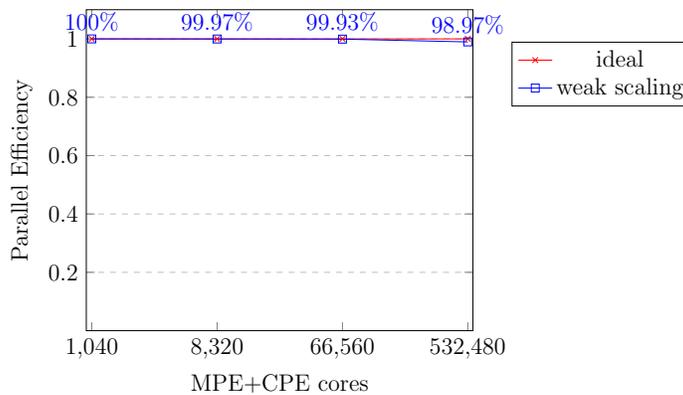
Results from both strong scalability and weak scalability indicate that our MISA-MD code have the ability to expand MD simulation to larger scale with large scale parallelism. Moreover, we also achieve 3.02×10^{13} atoms cascade collision simulation at 131 072 core groups on Sunway TaihuLight supercomputer. To our best knowledge, it holds the world record of the largest molecular dynamics simulation currently.

6. Conclusion

In this work, a new molecular dynamics code, MISA-MD, is developed for simulating material evolution in cascade collision. As a basic innovation, a new hash based data structure is proposed for efficient atoms storage and quick neighbor indexing. To fully utilize the performance of SW26010 processor, the EAM potential interaction computation is also implementation on Sunway TaihuLight with some efficient acceleration and optimization strategies, including a coloring method for avoiding writing conflict on sunway CPEs, double-buffer DMA and data reuse strategies etc. With the new data structure and accelerating of EAM interaction computation on SW26010 processor, we can extend MD simulation to larger temporal scale and longer spatial scales. It is worth noting that, many of our strategies are general



(a) Strong scalability for 6.55×10^{11} atoms with x axis logarithmic scaled. The number above the line is the parallel efficiency.



(b) Weak scalability for 8.39×10^6 atoms per core group (or MPI process) with x axis logarithmic scaled. The number above the line is the parallel efficiency.

Figure 10: Results of strong scalability and weak scalability performance tests.

and could also be implemented in other multi-core processors and heterogeneous platforms. For example, the coloring method for sunway CPEs can be easily migrated to OpenMP multiple threads program model or CUDA platforms. With these methods above, performance of 79.5% parallel efficiency is obtained in strong scaling simulation on Sunway TaihuLight system, and simulation of more than thirty-trillion atoms is performed using 131 072 sunway core groups. Moreover, experiments show that MISA-MD provides high accuracy of cascade collision simulation and has less memory overhead and higher performance compared with classical molecular dynamics code LAMMPS.

Moreover, MISA-MD is designed to couple its simulation result with kinetic Monte Carlo (kMC) method[37] and Cluster Dynamics (CD) method[38] for further multiscale evolution simulation [39, 40, 41, 42] of cluster and defects in long temporal scale(seconds to years).

Acknowledgment

This article is the results of the research project funded by The National Natural Science Foundation of China under Grant U1867217.

References

- [1] R. Stoller, Primary Radiation Damage Formation, in: Comprehensive Nuclear Materials, Elsevier, 2012, pp. 293–332. doi:10.1016/B978-0-08-056033-5.00027-6.
URL <https://linkinghub.elsevier.com/retrieve/pii/B9780080560335000276>
- [2] J. Knaster, A. Moeslang, T. Muroga, Materials research for fusion, Nature Physics (2016) 424–434doi:10.1038/nphys3735.
URL <http://www.nature.com/doifinder/10.1038/nphys3735>
- [3] K. Nordlund, S. J. Zinkle, A. E. Sand, F. Granberg, R. S. Averback, R. E. Stoller, T. Suzudo, L. Malerba, F. Banhart, W. J. Weber, F. Willaime, S. L. Dudarev, D. Simeone, Primary radiation damage: A review of current understanding and models, Journal of Nuclear Materials 512 (2018) 450–479. doi:10.1016/j.jnucmat.2018.10.027.
URL <https://linkinghub.elsevier.com/retrieve/pii/S002231151831016X>

- [4] K. Nordlund, Historical review of computer simulation of radiation effects in materials, *Journal of Nuclear Materials* 520 (2019) 273–295, 00000. doi:10.1016/j.jnucmat.2019.04.028.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0022311518314703>
- [5] Q. Peng, F. Meng, Y. Yang, C. Lu, H. Deng, L. Wang, S. De, F. Gao, Shockwave generates < 100 > dislocation loops in bcc iron, *Nature Communications* 9 (1), 00000. doi:10.1038/s41467-018-07102-3.
URL <http://www.nature.com/articles/s41467-018-07102-3>
- [6] J. Fu, Y. Chen, J. Fang, N. Gao, W. Hu, C. Jiang, H.-B. Zhou, G.-H. Lu, F. Gao, H. Deng, Molecular dynamics simulations of high-energy radiation damage in W and W–Re alloys, *Journal of Nuclear Materials* 524 (2019) 9–20, 00000. doi:10.1016/j.jnucmat.2019.06.027.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0022311519304416>
- [7] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, Tech. rep., Sandia National Labs., Albuquerque, NM (United States) (1993).
- [8] X. Duan, P. Gao, T. Zhang, M. Zhang, W. Liu, W. Zhang, W. Xue, H. Fu, L. Gan, D. Chen, X. Meng, G. Yang, Redesigning LAMMPS for Peta-Scale and Hundred-Billion-Atom Simulation on Sunway Taihu-Light, in: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, Dallas, TX, USA, 2018, pp. 148–159, 00000. doi:10.1109/SC.2018.00015.
URL <https://ieeexplore.ieee.org/document/8665774/>
- [9] C. Niethammer, S. Becker, M. Bernreuther, M. Buchholz, W. Eckhardt, A. Heinecke, S. Werth, H.-J. Bungartz, C. W. Glass, H. Hasse, J. Vrabec, M. Horsch, *ls1 mardyn* : The Massively Parallel Molecular Dynamics Code for Large Systems, *Journal of Chemical Theory and Computation* 10 (10) (2014) 4455–4464. doi:10.1021/ct500169q.
URL <https://pubs.acs.org/doi/10.1021/ct500169q>
- [10] J. Tersoff, New empirical approach for the structure and energy of covalent systems, *Phys. Rev. B* 37 (12) (1988) 6991–7000, publisher: Amer-

- ican Physical Society. doi:10.1103/PhysRevB.37.6991.
URL <https://link.aps.org/doi/10.1103/PhysRevB.37.6991>
- [11] J. E. Lennard-Jones, Cohesion, Proceedings of the Physical Society 43 (5) (1931) 461.
- [12] M. S. Daw, M. I. Baskes, Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals, Physical Review B 29 (12) (1984) 6443–6453, 00000. doi:10.1103/PhysRevB.29.6443.
URL <https://link.aps.org/doi/10.1103/PhysRevB.29.6443>
- [13] D. C. Rapaport, The art of molecular dynamics simulation, 2nd Edition, Cambridge University Press, Cambridge, UK ; New York, NY, 2004.
- [14] T. Law, J. Hancox, S. Wright, S. Jarvis, An algorithm for computing short-range forces in molecular dynamics simulations with non-uniform particle densities, Journal of Parallel and Distributed Computing 130 (2019) 1–11. doi:10.1016/j.jpdc.2019.03.008.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0743731519302047>
- [15] S. J. Plimpton, B. A. Hendrickson, Parallel molecular dynamics with the embedded atom method 291 37. doi:10.1557/PROC-291-37.
URL <http://link.springer.com/10.1557/PROC-291-37>
- [16] L. Zhigilei, Introduction to atomistic simulations, University of Virginia), MSE 4270 (2014) 6270.
- [17] H. Fu, J. Liao, J. Yang, L. Wang, Z. Song, X. Huang, C. Yang, W. Xue, F. Liu, F. Qiao, W. Zhao, X. Yin, C. Hou, C. Zhang, W. Ge, J. Zhang, Y. Wang, C. Zhou, G. Yang, The Sunway TaihuLight supercomputer: system and applications, Science China Information Sciences 59 (7). doi:10.1007/s11432-016-5588-7.
URL <http://link.springer.com/10.1007/s11432-016-5588-7>
- [18] G.-W. Yang, H.-H. Fu, Application software beyond exascale: challenges and possible trends, Frontiers of Information Technology & Electronic Engineering 19 (10) (2018) 1267–1272. doi:10.1631/FITEE.1800459.
URL <http://link.springer.com/10.1631/FITEE.1800459>

- [19] A homegrown many-core processor architecture for high-performance computing, SCIENTIA SINICA Informationis00006. doi:10.1360/N112014-00299.
URL <http://engine.scichina.com/doi/10.1360/N112014-00299>
- [20] H. J. Berendsen, D. van der Spoel, R. van Drunen, Gromacs: a message-passing parallel molecular dynamics implementation, Computer physics communications 91 (1-3) (1995) 43–56.
- [21] T. Zhang, L. Gan, H. Fu, W. Xue, W. Liu, G. Yang, Y. Li, P. Gao, Q. Shao, M. Shao, M. Zhang, J. Zhang, X. Duan, Z. Liu, SW_gromacs: accelerate GROMACS on Sunway TaihuLight, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '19, ACM Press, Denver, Colorado, 2019, pp. 1–14. doi:10.1145/3295500.3356190.
URL <http://dl.acm.org/citation.cfm?doid=3295500.3356190>
- [22] J. Phillips, Gengbin Zheng, S. Kumar, L. Kale, NAMD: Biomolecular Simulation on Thousands of Processors, in: ACM/IEEE SC 2002 Conference (SC'02), IEEE, Baltimore, MD, USA, 2002, pp. 36–36. doi:10.1109/SC.2002.10019.
URL <http://ieeexplore.ieee.org/document/1592872/>
- [23] L. V. Kale, S. Krishnan, CHARM++: a portable concurrent object oriented system based on C++, ACM SIGPLAN Notices 28 (10) (1993) 91–108. doi:10.1145/167962.165874.
URL <http://portal.acm.org/citation.cfm?doid=167962.165874>
- [24] N. Tchipev, S. Seckler, M. Heinen, J. Vrabec, F. Gratl, M. Horsch, M. Bernreuther, C. W. Glass, C. Niethammer, N. Hammer, B. Krischok, M. Resch, D. Kranzlmüller, H. Hasse, H.-J. Bungartz, P. Neumann, TweTriS: Twenty trillion-atom simulation, The International Journal of High Performance Computing Applications (2019) 109434201881974doi:10.1177/1094342018819741.
URL <http://journals.sagepub.com/doi/10.1177/1094342018819741>
- [25] C. Hu, H. Bai, X. He, B. Zhang, N. Nie, X. Wang, Y. Ren, Crystal MD: The massively parallel molecular dynamics software for metal with BCC structure, Computer Physics Communications 211 (2017) 73–78.

- doi:10.1016/j.cpc.2016.07.011.
URL <https://linkinghub.elsevier.com/retrieve/pii/S001046551630193X>
- [26] C. Hu, X. Wang, J. Li, X. He, S. Li, Y. Feng, S. Yang, H. Bai, Kernel optimization for short-range molecular dynamics, *Computer Physics Communications* 211 (2017) 31–40. doi:10.1016/j.cpc.2016.07.010.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0010465516301928>
- [27] T. C. Germann, K. Kadau, P. S. Lomdahl, 25 Tflop/s Multibillion-Atom Molecular Dynamics Simulations and Visualization Analysis on BlueGene/L 13.
- [28] L. Verlet, Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules, *Phys. Rev.* 159 (1) (1967) 98–103, publisher: American Physical Society. doi:10.1103/PhysRev.159.98.
URL <https://link.aps.org/doi/10.1103/PhysRev.159.98>
- [29] R. W. Hockney, S. P. Goel, J. W. Eastwood, Quiet high-resolution computer models of a plasma, *Journal of Computational Physics* 14 (2) (1974) 148 – 158. doi:[https://doi.org/10.1016/0021-9991\(74\)90010-2](https://doi.org/10.1016/0021-9991(74)90010-2).
URL <http://www.sciencedirect.com/science/article/pii/S0021999174900102>
- [30] J. Lin, Z. Xu, L. Cai, A. Nukada, S. Matsuoka, Evaluating the SW26010 many-core processor with a micro-benchmark suite for performance optimizations, *Parallel Computing* 77 (2018) 128–143. doi:10.1016/j.parco.2018.06.001.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0167819118301820>
- [31] G. Bonny, R. Pasianot, N. Castin, L. Malerba, Ternary Fe–Cu–Ni many-body potential to model reactor pressure vessel steels: First validation by simulated thermal annealing, *Philosophical Magazine* 89 (34-36) (2009) 3531–3546, 00000. doi:10.1080/14786430903299824.
URL <http://www.tandfonline.com/doi/abs/10.1080/14786430903299824>

- [32] S. Li, B. Wu, Y. Zhang, X. Wang, J. Li, C. Hu, J. Wang, Y. Feng, N. Nie, Massively Scaling the Metal Microscopic Damage Simulation on Sunway TaihuLight Supercomputer, in: Proceedings of the 47th International Conference on Parallel Processing - ICPP 2018, ACM Press, Eugene, OR, USA, 2018, pp. 1–11. doi:10.1145/3225058.3225064. URL <http://dl.acm.org/citation.cfm?doid=3225058.3225064>
- [33] L. Cai, Y.-C. Wang, W. Tang, B. Wang, S. Ethier, Z. Liu, J. Lin, OpenACC vs the native programming on sunway TaihuLight: A case study with GTC-p, in: 2018 IEEE International Conference on Cluster Computing (CLUSTER), IEEE, pp. 88–97. doi:10.1109/CLUSTER.2018.00021. URL <https://ieeexplore.ieee.org/document/8514862/>
- [34] Y. Liu, C. Hu, C. Zhao, Efficient parallel implementation of Ewald summation in molecular dynamics simulations on multi-core platforms, Computer Physics Communications 182 (5) (2011) 1111–1119. doi:10.1016/j.cpc.2011.01.007. URL <https://linkinghub.elsevier.com/retrieve/pii/S0010465511000270>
- [35] C. Hu, Y. Liu, J. Li, Efficient Parallel Implementation of Molecular Dynamics with Embedded Atom Method on Multi-core Platforms, in: 2009 International Conference on Parallel Processing Workshops, IEEE, Vienna, Austria, 2009, pp. 121–129. doi:10.1109/ICPPW.2009.24. URL <http://ieeexplore.ieee.org/document/5363184/>
- [36] M. Norgett, M. Robinson, I. Torrens, A proposed method of calculating displacement dose rates, Nuclear Engineering and Design 33 (1) (1975) 50–54. doi:10.1016/0029-5493(75)90035-7. URL <https://linkinghub.elsevier.com/retrieve/pii/0029549375900357>
- [37] A. F. Voter, INTRODUCTION TO THE KINETIC MONTE CARLO METHOD, in: K. E. Sickafus, E. A. Kotomin, B. P. Uberuaga (Eds.), Radiation Effects in Solids, Vol. 235, Springer Netherlands, Dordrecht, 2007, pp. 1–23. doi:10.1007/978-1-4020-5295-8_1. URL http://link.springer.com/10.1007/978-1-4020-5295-8_1

- [38] P. Terrier, M. Athènes, T. Jourdan, G. Adjanor, G. Stoltz, Cluster dynamics modelling of materials: A new hybrid deterministic/stochastic coupling approach, *Journal of Computational Physics* 350 (2017) 280–295. doi:10.1016/j.jcp.2017.08.015.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0021999117305867>
- [39] M. J. Caturla, R. K. Corzine, M. R. James, G. A. Greene, Multiscale modeling of radiation damage: applications to damage production by GeV proton irradiation of Cu and W, and pulsed irradiation effects in Cu and Fe, *Journal of Nuclear Materials* (2001) 11.
- [40] D. Molnar, R. Mukherjee, A. Choudhury, A. Mora, P. Binkele, M. Selzer, B. Nestler, S. Schmauder, Multiscale simulations on the coarsening of Cu-rich precipitates in α -Fe using kinetic Monte Carlo, molecular dynamics and phase-field simulations, *Acta Materialia* 60 (20) (2012) 6961–6971, 00000. doi:10.1016/j.actamat.2012.08.051.
URL <https://linkinghub.elsevier.com/retrieve/pii/S1359645412006003>
- [41] B. Wirth, G. Odette, J. Marian, L. Ventelon, J. Young-Vandersall, L. Zepeda-Ruiz, Multiscale modeling of radiation damage in Fe-based alloys in the fusion environment, *Journal of Nuclear Materials* 329-333 (2004) 103–111, 00000. doi:10.1016/j.jnucmat.2004.04.156.
URL <https://linkinghub.elsevier.com/retrieve/pii/S0022311504001321>
- [42] C.-C. Fu, J. D. Torre, F. Willaime, J.-L. Bocquet, A. Barbu, Multiscale modelling of defect kinetics in irradiated iron, *Nature Materials* 4 (1) (2004) 68–74. doi:10.1038/nmat1286.
URL <http://www.nature.com/doi/10.1038/nmat1286>

Algorithm 1: Iterate neighbor atoms to calculate force

Input: *arr_atoms* - instance of hash array of lattice list.

Input: *clash_atoms* - instance of hash clash of lattice list.

Input: $(x_{start}, y_{start}, z_{start}), (x_{end}, y_{end}, z_{end})$ - start and end lattice coordinate of the region for local atoms.

Input: *offset_list* - precalculated lattice id offset list

```
1 for  $x \leftarrow x_{start}, x_{end}; y \leftarrow y_{start}, y_{end}; z \leftarrow z_{start}, z_{end}$  do
2    $id \leftarrow 2 \times box_x \times box_y \times z + 2 \times box_x \times y + x$ ;
3    $atom \leftarrow arr\_atoms[id]$ ;
4   if  $atom$  is valid then
5     for every  $offset \in offset\_list$  do
6        $atom\_nei \leftarrow arr\_atoms[id + offset]$ ;
7       if  $atom\_nei$  is valid then
8          $f \leftarrow FORCE(atom, atom\_nei)$ ;
9          $atom.f \leftarrow atom.f + f$ ;
10  for every  $atom \in clash\_atoms$  do
11     $id \leftarrow HASH(atom.x, atom.y, atom.z)$ ;
12    if  $id$  specified lattice is not in ghost area then
13       $atom\_near \leftarrow arr\_atoms[id]$ ;
14       $f \leftarrow FORCE(atom, atom\_near)$ ;
15       $atom.f \leftarrow atom.f + f$ ;
16      for every  $offset \in offset\_list$  do
17         $atom\_nei \leftarrow arr\_atoms[id + offset]$ ;
18        if  $atom\_nei$  is valid then
19           $f \leftarrow FORCE(atom, atom\_nei)$ ;
20           $atom.f \leftarrow atom.f + f$ ;
21  for every  $atom \in clash\_atoms$  do
22     $id \leftarrow HASH(atom.x, atom.y, atom.z)$ ;
23    if  $id$  specified lattice is not in ghost area then
24      for every  $atoms\_cla \in clash\_atoms(id)$  do
25        if  $atom\_cla \neq atom$  then
26           $f \leftarrow FORCE(atom, atom\_cla)$ ;
27           $atom.f \leftarrow atom.f + f$ ;
28      for every  $offset \in offset\_list$  do
29         $id\_nei \leftarrow id + offset$ ;
30        for every  $atom\_cl \in clash\_atoms(id\_nei)$  do
31           $f \leftarrow FORCE(atom, atom\_cl)$ ;
32           $atom.f \leftarrow atom.f + f$ ;
```

Algorithm 2: Hash updating

Input: *arr_atoms* - instance of hash array of lattice list.

Input: *clash_atoms* - instance of hash clash of lattice list.

Input: $(x_{start}, y_{start}, z_{start}), (x_{end}, y_{end}, z_{end})$ - start and end lattice coordinate of the region for local atoms.

```
1 for  $x \leftarrow x_{start}, x_{end}$  do
2   for  $y \leftarrow y_{start}, y_{end}$  do
3     for  $z \leftarrow z_{start}, z_{end}$  do
4        $id \leftarrow 2 \times box_x \times box_y \times z + 2 \times box_x \times y + x$ 
5        $atom \leftarrow arr\_atoms[id]$ 
6       if  $atom$  is invalid then
7         continue
8        $id\_real \leftarrow HASH(atom.x, atom.y, atom.z)$ 
9       if  $id \neq id\_real$  then
10        inset  $atom$  into clash_atoms
11        set flag of  $arr\_atoms[id]$  as invalid
12 for every  $atom \in clash\_atoms$  do
13    $id \leftarrow HASH(atom.x, atom.y, atom.z)$ 
14   if  $id$  specified lattice is not in ghost region then
15     if  $arr\_atoms[id]$  is invalid then
16       set flag of element  $arr\_atoms[id]$  as valid
17        $arr\_atoms[id] \leftarrow atom$ 
18       remove  $atom$  from clash_atoms
```

Table 1: Speedup of MISA-MD on single core group with and without CPEs.

	MPE only	MPE+64CPEs
Execution time (seconds)	1192.33	21.2939
Speedup	1	55.99

Table 2: Memory Usage of MISA-MD Compared with LAMMPS on TianHe-2

No.	Program	Cores	Box	Mem usage (GiB/core)	Mem per atom (Bytes/atom)
1	MISA-MD	120	600 ³	0.64	191.2
2	LAMMPS	120	600 ³	1.77	527.2
3	MISA-MD	120	1094 ³	2.66	130.9
4	LAMMPS	120	668 ³	2.66	574.9