Computational Materials Science 144 (2018) 1–10

Contents lists available at ScienceDirect

Computational Materials Science

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

A spatial decomposition parallel algorithm for a concurrent atomistic-continuum simulator and its preliminary applications

Hao Chen [a,*], Shuozhi Xu [b], Weixuan Li [c], Rigelesaiyin Ji [a], Thanh Phan [a], Liming Xiong [a]

[a] Department of Aerospace Engineering, Iowa State University, Ames, IA 50010, United States
[b] California NanoSystems Institute, University of California, Santa Barbara, CA 93106-6105, United States
[c] Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32608, United States

ARTICLE INFO

Article history:
Received 31 August 2017
Received in revised form 28 November 2017
Accepted 29 November 2017

Keywords:
Multiscale
Concurrent atomic continuum simulation
Parallel algorithm
Efficiency

ABSTRACT

This paper presents the development of a spatial decomposition parallel algorithm and its implementation into a concurrent atomistic-continuum (CAC) method simulator for multiscale modeling of dislocations in metallic materials. The scalability and parallel efficiency of the parallelized CAC are tested using up to 512 processors. With a modest computational resource, a single crystalline f.c.c. sample containing 10.6 billion atoms is modeled using only 4,809,108 finite elements in a CAC model at a fraction of the cost of full molecular dynamics (MD). The simulation demonstrates a nearly ideal scalability of the newly parallelized CAC simulator. The parallel efficiency of the newly parallelized CAC is shown to be higher than 90% when using 512 processors in the high performance computing cluster at Iowa State University. This parallel efficiency is comparable to the state-of-the-art atomistic simulator. Moreover, the newly parallelized CAC simulator employing a uniform coarse mesh is capable of capturing important atomistic features of dislocations, including dislocation nucleation, migration, stacking faults as well as the formation of Lomer-Cottrell locks, in a billion-atom system. The spatial decomposition-based parallelization algorithm developed in this work is general and can be transferable to many other existing concurrent multiscale simulation tools.

1. Introduction

Many material behaviors and engineering processes are multiscale in nature. Understanding those behaviors and processes across a broad range of length scales are important for the development of novel engineering structures and materials. For example, the plastic deformation of metallic materials spans a wide range of lengths scales ranging from dislocation nucleation at the atomic scale to the formation of multiple slip bands, planar dislocation arrays, and dislocation cells at the microscale, and to the observable effect of permanent deformation at the macroscopic level [1]. It is believed that the multiscale nature of plasticity precludes direct simulations using a formulation appropriate only for one single length scale [2]. For instance, a fully atomistic simulation can provide atomistic details of plastic deformation, such as dislocation interactions, dislocation networks [3,4]. However, it requires a formidable computational cost if a prediction of the macroscopic-level plastic deformation is desired. By contrast, the continuum-level theoretical and computational framework, such as crystal plasticity finite element method (CPFEM), are applicable to simulating the plastic behavior in materials at the macroscopic level. Nevertheless, these approaches lack a predictive capability from the bottom up because they ignore the atomistic discrete nature of materials. In the past decades, taking the advantage of atomistic simulations and continuum-level methods, extensive efforts have been dedicated to the development of multiscale methods for modeling plasticity [5]. Existing multiscale methods generally fall into two categories: sequential and concurrent approaches. In sequential methods, MD simulations are deployed to calibrate the constitutive models and parameters for higher order models. For example, to simulate material plasticity, information about dislocation nucleation, the strength of dislocation junctions, dislocation mobility, and dislocation interactions from atomistic modelling are used to develop short-range interaction rules. These rules are then feed into continuum-level models such as dislocation dynamics [6–8]. One major challenge of such sequential approach is how to average the fine scale information and how to input the averaged information into the higher scale models. In contrast, concurrent methods directly combine a fine-scale description of materials with a higher order material description within one computer model [9–14]. Examples of such

* Corresponding author.
E-mail address: haochen@iastate.edu (H. Chen).

https://doi.org/10.1016/j.commatsci.2017.11.051
0927-0256/

concurrent methods for multiscale plasticity include the coupled atomistic and discrete dislocation formulation [15–19], coupled discrete dislocation and continuum crystal plasticity [2], and the multiresolution molecular mechanics [20]. A comprehensive review of concurrent multiscale methods can be found in Refs. [21,22].

A direct combination of a fine-scale model, e.g., MD, with a higher order model, such as finite element (FE), within one computational framework introduces an unrealistic numerical interface into the computational model. Due to the mismatch of material descriptions between MD and FE, an atomistic/continuum interface needs to be constructed through a careful numerical implementation. Most of the efforts are then devoted to construct such numerical interface by matching or bridging the atomistic and continuum descriptions such that defects can pass from MD to FE. Different from other concurrent multiscale methods which directly combines MD and FE, the recently developed concurrent atomistic-continuum (CAC) method is based on a new atomistic field formalism that unifies the atomistic and continuum description of materials within one theoretical framework [23]. The coarse-grained (CG) domain in CAC admits dislocation nucleation and migration on the boundaries within the gaps between elements without the need of adaptive mesh refinement [23]. CAC has been successfully applied to simulate a variety of material behaviors, such as slip transfer of dislocations across grain boundaries [24,25], dislocations bowing out from Frank-Read sources [26], dislocation-void interaction [27], dislocations and fracture in strontium titanate [28,29], dynamic crack propagation [30], fast moving dislocations [31], and phonon dynamics in a 1-D polyatomic chain [32]. It is noted that, because the original version of the CAC simulator was not massively parallelized, those applications were mainly limited to material behavior from the atomistic to the nanoscale.

In order to perform large-scale CAC simulation of material behaviors using hundreds of processors, this work aims to develop a massively parallelized CAC simulator that can be applied to predict dislocation activities in a submicron-sized specimen without smearing out its atomistic nature. A spatial decomposition (SD) algorithm is developed and implemented. This algorithm takes full advantage of the local nature of the SD procedure and shows a good parallel efficiency without any limitation on the number of processors. The remainder of this paper is organized as follows. In Section 2, the formulation and the algorithm of the CAC method are briefly reviewed. In Section 3, the SD algorithm as well as its efficiency and scalability are presented in details. In Section 4, simulations of dislocation activities in a single-crystalline face-centered cubic (f.c.c.) sample with the same computational setup as that in Ref. [23] is presented to benchmark the newly parallelized CAC. In Section 5, the scalability of the algorithm is demonstrated and compared against the fully atomistic simulator, the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) [33]. In Section 6, CAC simulations of the activities of submicron-long dislocation lines in a sample containing billions of atoms are presented. This paper ends with a brief summary and discussion.

2. A brief review of the CAC method

The theoretical foundation of CAC is an atomistic field formalism proposed by Chen [34,35], in which a crystalline material is viewed as a continuous collection of lattice points, while embedded within each point is a unit cell containing a group of discrete atoms [36]. Chen [34,35] defined the continuum-level physical quantities from the atomic scale and formulated the microscopic balance equations of the physical quantities including mass density, linear momentum density, and the internal energy density.

It is noted that, the continuum description of those physical quantities is by means of continuous functions in terms of $\boldsymbol{x}$ and t in the physical space. Microscopic dynamic quantities in classical N-body dynamics, on the other hand, are functions of $(\boldsymbol{r}, \boldsymbol{p})$, i.e., the positions and momenta of atoms, in phase space:

$$\boldsymbol{r} = \{\boldsymbol{R}^k, k = 1, 2, 3, \ldots, n\},$$

$$\boldsymbol{p} = \{m^k \boldsymbol{V}^k, k = 1, 2, 3, \ldots, n\}. \tag{1}$$

where $\boldsymbol{R}^k$ is the position vector and $\boldsymbol{V}^k$ is the velocity of the k th atom, m^k is the atomic mass, and n is the total number of atoms in the system. In the atomistic field formalism, the quantities in the phase space and the physical space descriptions can be linked through utilizing the localization function δ [34–36], i.e.,

$$\rho(\boldsymbol{x}) = \sum_{k=1}^{n} m^k \delta(\boldsymbol{R}^k - \boldsymbol{x}), \tag{2}$$

$$\rho(\boldsymbol{x})\boldsymbol{v}(\boldsymbol{x}) = \sum_{k=1}^{n} m^k V^k \delta(\boldsymbol{R}^k - \boldsymbol{x}), \tag{3}$$

where ρ is the microscopic local mass density and $\rho\boldsymbol{v}$ is the momentum density in physical coordinate. Taking the time derivatives of Eqs. (2) and (3), Chen [34,35] formulated the microscopic balance equations of the mass and linear momentum. In particular, for a monoatomic crystal under no external forces, the balance equation of linear momentum can be re-written as [23]

$$\rho\ddot{\boldsymbol{x}} = \boldsymbol{f}_{int}(\boldsymbol{x}) \tag{4}$$

where

$$\boldsymbol{f}_{int} = \sum_{k=1}^{n}\sum_{i} \delta(R^k - x) F_{ik} \tag{5}$$

is the internal force density, F_{ik} is the atomic force acting on atom k by atom i and δ is a Dirac δ -function.

It is daunting to directly compute F_{ik} for all the atomic pairs. Thus, one critical step in numerical implementation of the atomistic field formalism is to efficiently and accurately calculate the internal force density $\boldsymbol{f}_{int}$ in Eq. (5). Xiong et al. [23] performed a FE implementation of the atomistic field formalism and the numerical procedure was coined as CAC, which discretizes the computational domain into piecewise elements. Gaussian quadrature was deployed to perform the spatial integration to calculate the internal force density [23]. Later, Yang et al. [28] and Xu et al. [24] modified the quadrature rules and demonstrated that the generalized stacking fault energy, core structure and the stress field around a mixed dislocation in CAC models are comparable with that from MD. In this paper, the integration scheme in Ref. [24] is employed. A computational domain is discretized into piecewise elements (Fig. 1). Each element is in a rhombohedral shape and contains a collection of lattice cells. The displacement within the element is approximated by

$$\boldsymbol{u} = \Phi_\xi(\boldsymbol{x})\boldsymbol{U}_\xi \tag{6}$$

where $\boldsymbol{U}_\xi$ are finite element nodal displacements, $\Phi_\xi(\boldsymbol{x})$ is the standard tri-linear shape function. In the numerical integration scheme, the internal force on node ξ is calculated as follows [23,24]:

$$f_{int}^{\xi} = \frac{\sum_\mu \omega_\mu \Phi_{\xi\mu} F^\mu}{\sum_\mu \omega_\mu \Phi_{\xi\mu}} \tag{7}$$

where F^μ is the force on the integration point μ, ω_μ is the weight, and $\Phi_{\xi\mu}$is the value of the shape function Φ_ξ at the integration point μ. The weight ω_μ is determined by the number of atoms that the

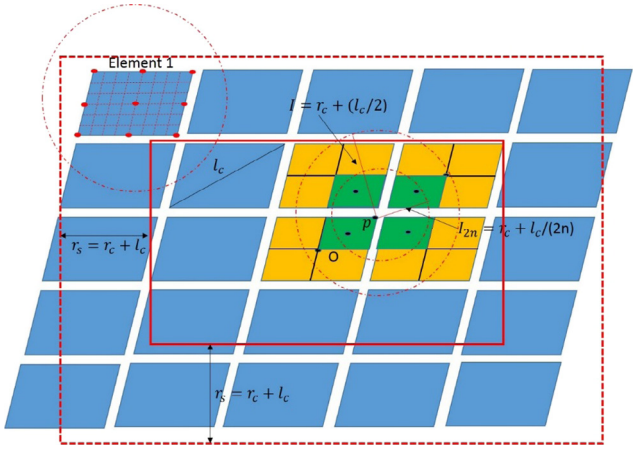

Fig. 1. The schematic sketch for building the neighbor list of an integration point p in an element centered at point O. The domain bounded by the solid red lines is referred as a "local box" handled by a local processor. The domain bounded by the dotted red line is referred as a "ghost box". The information associated with the elements (elements in blue) falling into the "ghost box" but out of the "local box" is sent to the local processor. Here l_c is the element size which is the largest length of the diagonals and r_c is the cutoff of the interatomic potential. The domain in yellow covers the atomic interaction range associated with an integration point p. In particular, within this interaction range, the instantaneous position of each atom falling into the domain in green is interpolated from the FE nodal positions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

integration point represents. In details, for those integration points located on nodal sites, $\omega_\mu = 1$; for those on the element edges, $\omega_\mu = N_l - 2$; for those on the surface, $\omega_\mu = (N_l - 2)^2$; and for those within the interior of the elements, $\omega_\mu = (N_l - 2)^3$. Here, N_l is the number of atoms along the edge of each coarse element. It is clear that the CAC model will degenerate into full MD simulations when $N_l = 2$ *and* $\mu = 8$. Such an atomic-scale mesh is referred as the finest mesh in CAC method and will function as a fully atomistic domain. In this paper, uniform coarse elements are used to discretize the material sample and the number of integration points within each element is set as $\mu = 27$, which corresponds to the '1NN' case in Ref. [24]. For $N_l \leqslant 3$, a collocation integration procedure is used and $\mu = 8$.

3. Parallelization algorithm

The CAC simulator is parallelized through the Message Passing Interface (MPI) using the spatial decomposition algorithm. The main idea of this algorithm is to evenly divide the total volume of the system into small boxes with equal volume, e.g., the red solid box in Fig. 1. Each small box is assigned to a processor. Specifically, we use a regular 3-dimensional meshing topology [33] in which the simulation box is divided into D_α parts along $\alpha(\alpha = x, y, z)$ direction. The total number of processors is $p = D_x D_y D_z$ and each processor is indexed as $d = d_x D_y D_z + d_y D_z + d_z$. The sub-box owned by the d-th processor is indexed as d_α along the αdirection. Element i with the center position $\boldsymbol{r}_{ic} = (r_{ix}, r_{iy}, r_{iz}) = \frac{1}{m}\Sigma \boldsymbol{r}_{in}$ is mapped to processor $d(\boldsymbol{r}_i)$ in an array. The computational domain along each direction is subdivided into D_x, D_y and D_z in an equal size

$$\begin{cases} d(\boldsymbol{r}_{ic}) = d_x(r_{ix})D_y D_z + d_y(r_{iy})D_z + d_z(r_{iz}), \\ d_\alpha(r_{i\alpha}) = \left[\frac{r_{i\alpha} D_\alpha}{L_\alpha}\right] \qquad (\alpha = x, y, z), \end{cases} \tag{8}$$

in which L_α is the simulation box size along the α direction. The box belonging to each processor is called the "local box" of that processor in comparison with the enlarged "ghost box" introduced later.

Each processor computes and updates the forces, positions and velocities of FE nodes within its own domain at each time step. Elements moving outside of the box will be assigned to the new box during the simulations. The information of elements in the neighboring boxes is sent to the local box through *MPI_Send* and *MPI_Irecv* command, i.e., the information of some elements in the red dotted box inside the nearby processors is sent to the local processor for computing the force on the nodes of the local elements. Here an enlarged box is defined and referred as a "ghost box" [33] which has all dimensions larger than the local box of l_c (Fig. 1). It should be pointed out that the data structure and the communication in the spatial decomposition is local in nature. This will in principle lead to a high parallel efficiency if each processor contains a similar number of elements. In contrast, in other parallel algorithms, such as the force decomposition method [33,37,38], globalized vectors are used and the communication of the globalized vectors (all-to-all) is required. As a consequence, the simulation employing such force decomposition becomes significantly slow when the number of degree of freedom increases. This leads to a decrease of the parallel efficiency in large-scale simulations [33].

In each processor, three dimensional arrays are constructed for storing the positions, velocities and forces of FE nodes. The first dimension of each array is the number of elements, the second is the number of nodes per element (8 for 3D and 4 for 2D), and the third is 3 for 3D or 2 for 2D, respectively. Nodal positions of elements with their center positions falling into the ghost box are sent from the neighboring processors and updated at each time step. Those nodal positions received from neighboring boxes are directly inserted at the end of the local vectors without sorting. The lists of elements received from the nearby processors are recorded in the following several time steps. The neighbor lists will be only updated when any of the FE nodal displacement is larger than a skin parameter. Such a technique has been employed in the MD simulator LAMMPS [33]. The positions of the atoms within each element are interpolated using the shape functions in Eq. (6) [23–24].

The parallelization starts with reading the input data file. Fig. 2 shows that the FE nodal coordinates are read by the master processor, noted as CPU_0, step by step to avoid creating global data structures in all of processors. At each step, CPU_0 reads a fixed number of element coordinates: l_n. In this work, we choose l_n as 1024 in order to balance the input reading and the information communication between processors. Thereafter, CPU_0 will broadcast the data to all the other processors, noted as *CPUs*, in the communication *world* using the *MPI_Bcast* command. In each processor, a vector of the same size of l_n is allocated in order to receive data broadcasted by CPU_0. With this information, each processor will determine whether the position of the element center, r_{ic}, falls into its own local box. Those elements falling into the local box will be assigned to the processor and the information associated with them is stored at the end of the existing position vectors. In this way, only local data structures will be needed. The same strategy is applied to the output of results during the simulation.

Fig. 3 shows the communication scheme between processors [33]. This strategy is the same as what has been implemented in LAMMPS because each element can be considered as a "coarse atom" [39]. Here the elements received from nearby processors are noted as ghost elements. The cutoff for ghost elements is set as $r_s = r_c + \max(l_c)$. Firstly, the positions of FE nodes falling within the cutoff length r_s of CPU_1 's box is sent to CPU_2 as shown in Fig. 3a followed by the reverse communication. The same procedure is repeated along the north/south direction in Fig. 3b. The only difference is that messages sent to the adjacent *CPU* now contain not only local elements but also ghost elements received from previous

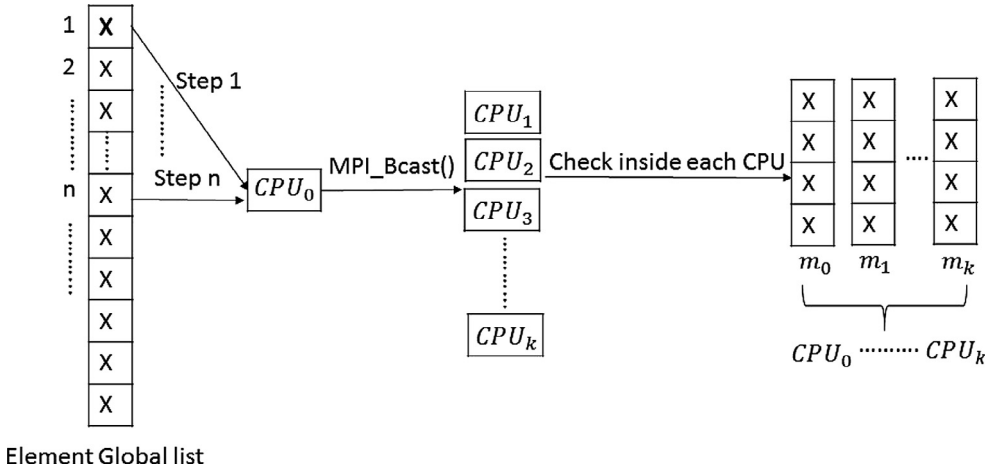

Fig. 2. The process of reading input file and the information broadcasting to all the processors.

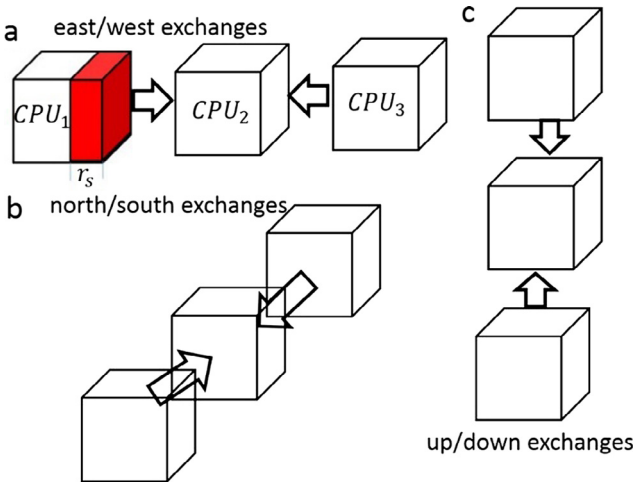

Fig. 3. A scheme of the communications between processors in the parallelized CAC simulator.

communications. This process is then repeated along the up/down dimension. When $r_s > L_\alpha/D_\alpha$, here L_α and D_α have the same meaning as that in Eq. (8), those elements in $\left[\frac{r_s D_\alpha}{L_\alpha}\right] + 1$ neighboring boxes are needed. Thus the communication procedure will be performed for $\left[\frac{r_s D_\alpha}{L_\alpha}\right] + 1$ times. The advantage of this scheme is to minimize the communication data [33] although the coding process becomes tedious. That is, each processor acquires only the elements that are within a distance r_s of its "local box". All the received elements are placed as contiguous data into the local data structure without rearranging. During the communication, a full scan of the data structure is only conducted when there is a need to decide the information of which element should be sent. Such a scan creates a list of elements that compose of each message. During all the other timesteps, the lists can be used to directly sort the referenced elements and buffer up the messages in an efficient manner.

Then the neighbor list of integration points is constructed for the evaluation of the internal force density. Once the neighbor list is constructed, the internal force density associated with each integration point can be calculated. The neighbor list construction in CAC is mainly divided into two steps. The first step is to build the element neighbor list using a link-cell scheme. The cutoff for the element neighbor list is the same as that used for communication: r_s. The bin size used here is $r_s/2$ which maximizes the efficiency of the neighbor list construction [33]. The second step is to build the neighbor list of integration points through searching the atoms interpolated from the neighboring elements. Since each element contains several thousands of atoms, interpolation of all atoms inside all elements will be demanding. Here, instead of performing a full-domain interpolation, the second step is further divided into two sub steps. Before going into details, we introduce one parameter: $I = r_c + l_c/2$, referred as the influence radius of an element (Fig. 1). The distance between the atoms outside this radius and the atoms interpolated from this element is larger than r_c and thus there will be no interaction between them. The process for finding the integration point i in Fig. 1 is described as following. Firstly, the distance between i and the center position of each neighboring element will be calculated. If the distance is larger than I, the atoms inside this element will not be interpolated. For example, the distance between the centroid of element 1 and the integration point p in Fig. 1 is larger than I. This means that the distance between any of atoms within element 1 and the integration point p will be larger than r_c. In this situation, all the atoms within element 1 will not be in the neighbor list of point p and will not be interpolated. Under this treatment, only the positions of the atoms falling into the elements in yellow (Fig. 1) will be interpolated for constructing the neighbor list of the integration point. The next sub-step is to further subdivide each yellow element in Fig. 1 into n equal-sized segments along each direction and n^2 (n^3 for 3D problems) smaller sub-elements in total. Each sub-element has an influential radius: $I_{sub} = r_c + l_c/(2n)$. When the distance between the sub-elements and the integration points is smaller than I_{sub}, the positions of the atoms will be interpolated for neighbor list construction. As such, the interpolation in the neighbor elements will be only performed in those sub-elements in green. Fig. 4 shows that the neighbor list construction time, t_n, changes with the splitting number n along each direction. The testing computer model used to produce the data in Fig. 4 contains 10.6 billion atoms and is discretized into 31,45,963 coarse elements [23]. Each element contains 3,375 atoms. Using 512 processors, the minimum neighbor building time is 2.1 seconds for this model. It is found that t_n first decreases at an order of 3, which is the same as the number of elements to be interpolated, and then increases slightly with the increase of the splitting number n. The increase of the neighbor list construction time is due to the time spent for checking the distances between integration points and sub-elements when the splitting number n become larger. With this information, an optimized value for element splitting can be determined for the neighbor list construction.

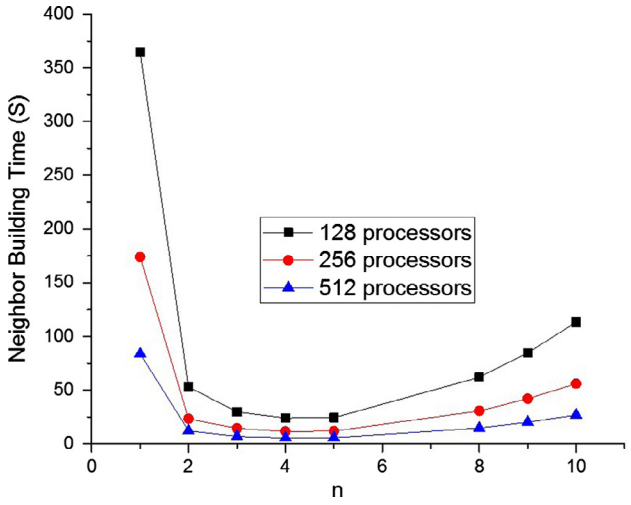

Fig. 4. The relation between the splitting number n and the time for the neighbor list construction.

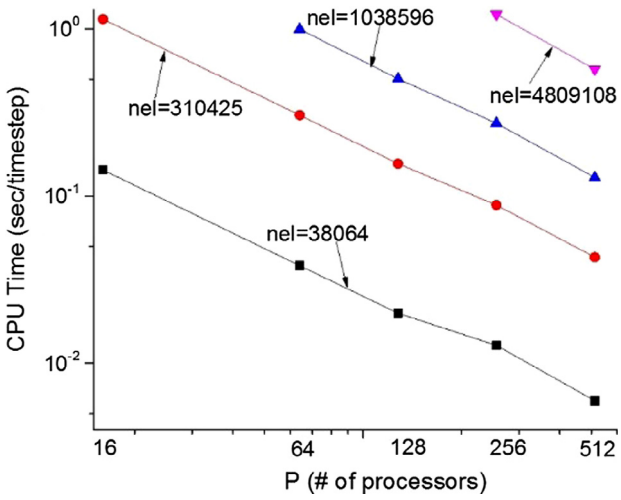

Fig. 6. CPU timing (seconds/timestep) versus the number of processors used for different CAC computer models containing different number of elements.

4. Scalability of the massively parallelized CAC simulator

To investigate the scalability of the newly developed parallel algorithm, CAC models for cubic f.c.c. samples with a dimension of $0.1\,\mu m \times 0.1\,\mu m \times 0.1\,\mu m$ and $0.5\,\mu m \times 0.5\,\mu m \times 0.5\,\mu m$ are tested. The CAC model for a cuboid sample ensures that each processor contains similar number of coarse elements and avoids unequal workload among different processors. The $0.5\,\mu m \times 0.5\,\mu m \times 0.5\,\mu m$ sample as shown in Fig. 5 is the largest computer model explored in this work. This model contains 10.566 billion atoms and is discretized into $4,809,108$ coarse elements. Each element contains 2197 atoms. The timestep is set as 5 fs. Fig. 6 shows the relationship between the number of processors and the averaged time cost for each timestep in different models containing different number of elements. The time cost is averaged over the total simulation time which includes the reading, initialization, the neighbor list construction and also the FE nodal force calculation. The computing timing of CAC using a single processor is taken as a reference point for calculating the parallel efficiency when memory for single processor is large enough to store all global vectors. It should be noted that, for problems when *nel* (number of elements) $= 4,809,108$, the memory associated with one single processor is not enough and simulations should start with a certain number of processors (for example, 16 for $nel = 310,425$). The reference points are then the timings of the starting number of processors. While the number of processors p increases, the size of the domain covered by the "local box" decreases. This leads to a higher volume ratio between the ghost box and the local box. That is, the ratio between the number of the elements containing in the ghost box and that in local box: R, increases. For example, for $nel = 38,064$ in Fig. 6, $R = 1$ when $P = 32$ and $R = 10$ when $P = 512$. This will induce a small decrease of the parallel efficiency. However, the spatial domain decomposition algorithm still retains a parallel efficiency of 85% when $P = 512$. When $nel \geqslant 310,425$ in Fig. 6, a parallel efficiency of 92% can be achieved when 512 processors are used. This is the maximum number of processors we can access using our computing resource. Nevertheless, since all data sets and communications are localized in the SD algorithm, the parallel efficiency remains high when $P > 512$ and will be comparable to that in MD simulations using LAMMPS [33].

Fig. 7 presents the linear relationship between *nel* and the total *CPU* time, $p * t_{su}$. Here t_{su} is the time spent by each processor. This linear relationship means that the total computational time is determined by the total number of elements. For all the cases in Fig. 7, the communication time doesn't exceed 8% of the total simulation time and more than 90% of the simulation time is spent in

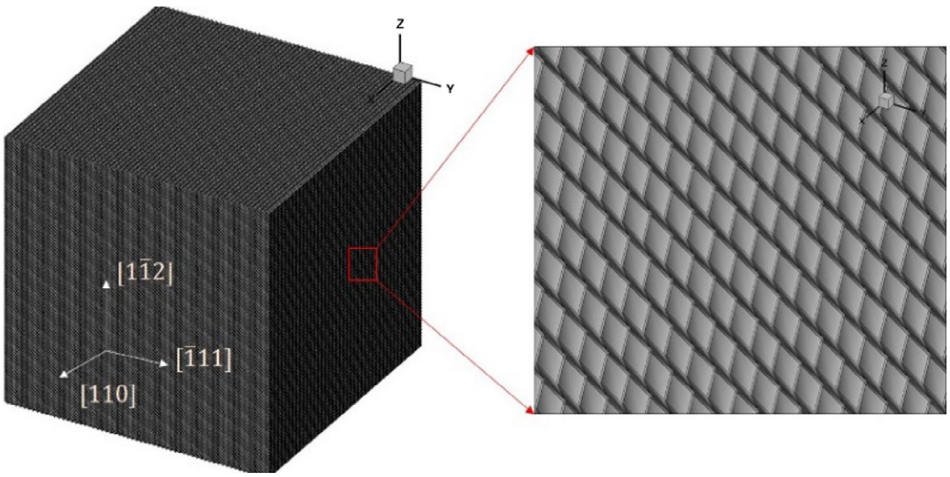

Fig. 5. CAC model of a 0.5μm cubic sample used to test the scalability.

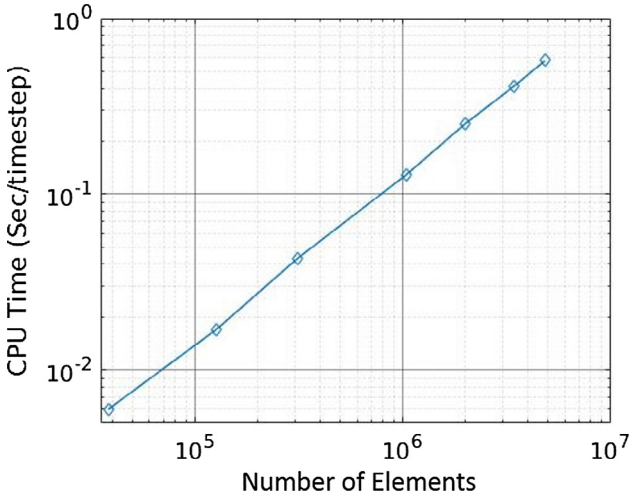

Fig. 7. A relationship between the CPU timing and the number of elements assigned to each processor.

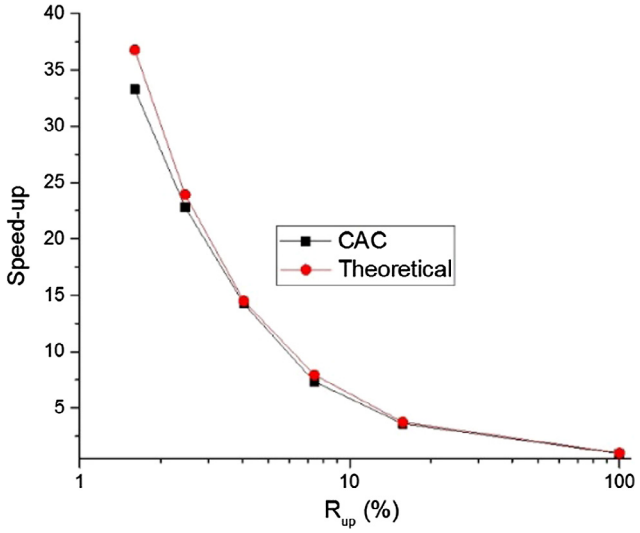

Fig. 8. The speed-up of the CAC model with respect to computational workload ratio (R_{up}).

the interpolation of the integration points, calculation of nodal force and updating nodal positions. This implies that there will be no limit of the size of models as long as we can have access to a large number of processors [33]. The high parallel efficiencies ensures that the SD algorithm presented in this work can be at the same level as that in many existing massively parallelized MD simulator [33].

5. Comparison with LAMMPS

In addition to the CAC computer model, the algorithm can also be applied to perform MD simulations when the coarse mesh in the CAC model is reduced to the atomic scale [23]. In this situation, a comparison between the CAC simulation time and LAMMPS [33] using exactly the same hardware [40] is carried out. In a 1000-timestep simulation of a $(0.1\,\mu m)^3$ cubic sample containing $83,626,608$ atoms using 16 nodes in the Condo cluster at Iowa State University [41], LAMMPS took 454.8 s and the CAC took 480.2 s. For a uniform coarse mesh in CAC, the computational workload ratio between the coarse-grained model and MD simulations can be defined as below

$$R_{up} = \frac{total\ number\ of\ integration\ points}{total\ number\ of\ atoms} \tag{9}$$

Here the theoretical speed-up is defined as $S_{theory} = 1/R_{up}$. The actual speed-up of the parallelized CAC with respect to various R_{up}, closely follows the theoretical speed-ups in general, as shown in Fig. 8. The largest speed-up we obtain is 33.3 when the uniform coarse element (2197 atoms per element) is used in CAC.

6. A simple validation

CAC computer models of notched single-crystal copper specimens in a previous work [23] are used to validate the correctness of the newly implemented parallel algorithm in CAC. A Lennard-Jones (L-J) potential is used, with parameters $\epsilon_0 = 0.167ev$ and $\sigma_0 = 2.3151$Å. This model contains $1,423,107$ atoms and is discretized into $4,149$ elements with each element containing 343 atoms. Fig. 9 presents the atomic rearrangements, dislocation structures, and stacking faults from the newly parallelized CAC and also from MD. Two dislocations are emitted from the notch tips and propagate into the specimen interior along {111} planes. The same non-symmetric dislocation nucleation behavior is captured here [23]. Also, the Lomer-Cottrell or stair-rod locks [41] are formed, which hinders further dislocation glide on the two slip planes and provides a barrier to other dislocations [23]. This is also found in MD simulations [23]. We test this model with random number of processors and results are the same. This demonstrates that the results are independent of the number of processors and the communications between processors are reliable in the massively parallelized CAC code.

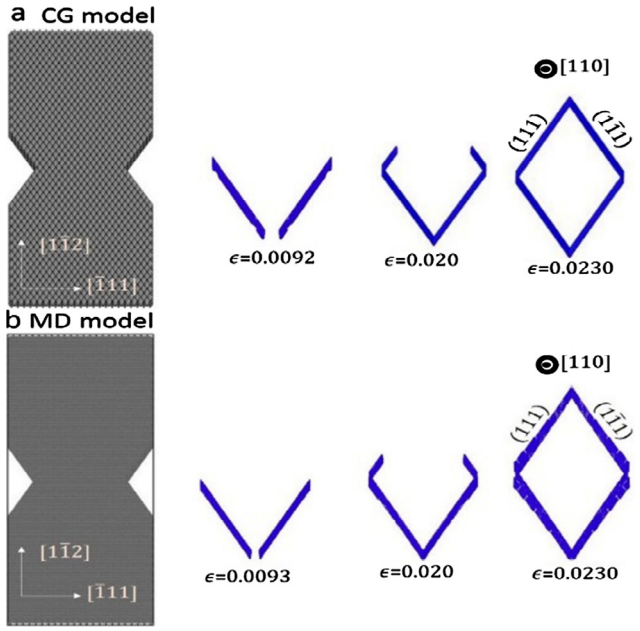

Fig. 9. Snapshots of atomic arrangements, dislocations and stacking faults. (a) Results from the newly parallelized CAC simulation. (b) Results from MD simulation. Here only the atoms associated with dislocations and stacking faults are displayed in blue. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

7. Numerical Examples

CAC furnished with the newly developed spatial decomposition algorithm is used to simulate dislocations in a $0.25\mu m * 0.25\mu m * 0.53\mu m$ cubic sample with two notches on

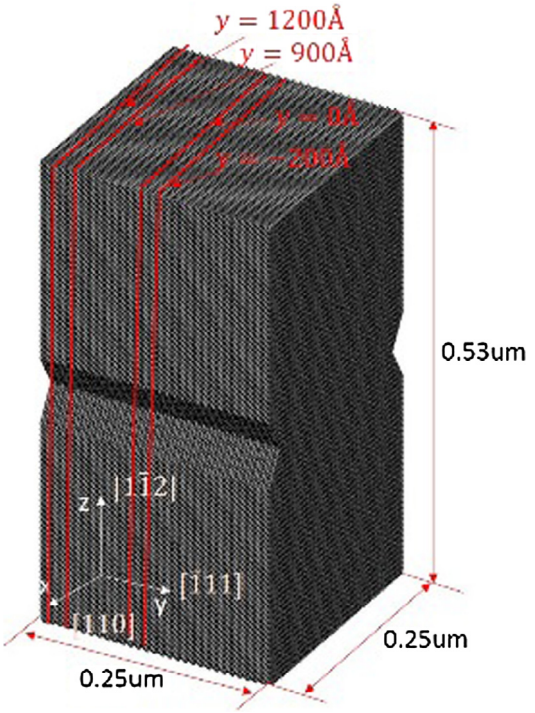

Fig. 10. CAC model for a billion-atom system to benchmark the spatial decomposition algorithm.

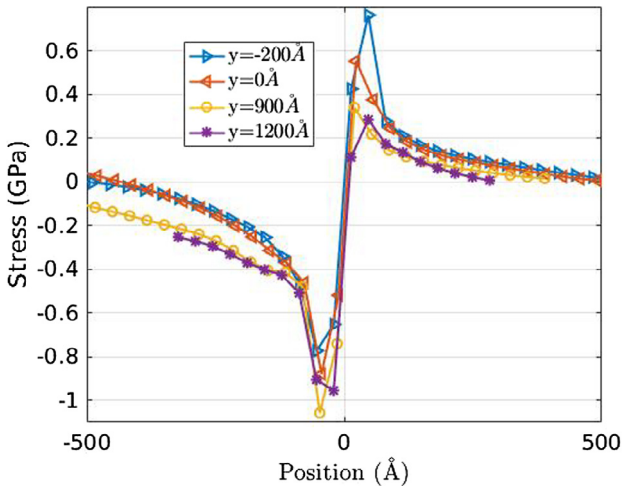

Fig. 12. Shear stress (σ_{xz}) field around the cores of dislocation in different cutting planes.

both sides as shown in Fig. 10. This model contains $2,307,616,753$ atoms and is discretized into $1,050,349$ elements with each element containing 2197 atoms. A constant velocity of $1m/s$ is applied (corresponding to a strain rate on the order of $10^6 s^{-1}$) on the two vertical (z direction) ends of the sample with the other surfaces traction free.

Fig. 11 shows that dislocations nucleate around the notches [42]. Through the Burgers vector analysis using the geometric method in OVITO [4], the emitted dislocations are found to be partial dislocations with Burgers vector $\frac{1}{6}[11\bar{2}]$ in the (111)-plane and $\frac{1}{6}[\bar{1}2\bar{1}]$ in the $(\bar{1}11)$-plane. Dislocations migrate in the sample with different velocities due to the free surface and elastic interactions between dislocations [4]. During their migrations, the curvature of the dislocation lines decreases to reduce the line energy of dislocations [43]. Eventually, the two dislocations lines interact with each other as shown in Fig. 11e. The interaction angle was measured as 18.2°, which remains constant until the two dislocations formed a Lomer-Cottrell lock [41].

It should be noted that dislocation lines along the thickness direction, i.e., the y-direction, insects two free surfaces in the xz plane. In order to characterize the effects of free surfaces on the dislocation core stress field, the shear stress distribution around the dislocation core at $y = -200, 0, 900 and 1200$Å is quantified and presented in Fig. 12. It is seen that the positive shear stress component decreased when the dislocation core is approaching the free surface. That is, the maximum positive shear stress decreases from $\sim 0.8\,GPa$ when $y = -200$Å to $\sim 0.2\,GPa$ when $y = 1200$Å. Clearly, the stress around a dislocation core gradually decays when it gets closer to a free surface. This result implies that dislocation core stress field in a finite-sized material sample is actually size-dependent and will be only approaching the analytical solution of a dislocation core embedded within an infinite media when the core is reasonably far away, e.g. $120\,nm$ in this work, from a free surface.

In order to demonstrate that the applicability of the newly parallelized CAC code to different material systems in which use different interatomic potentials, one set of massively parallelized

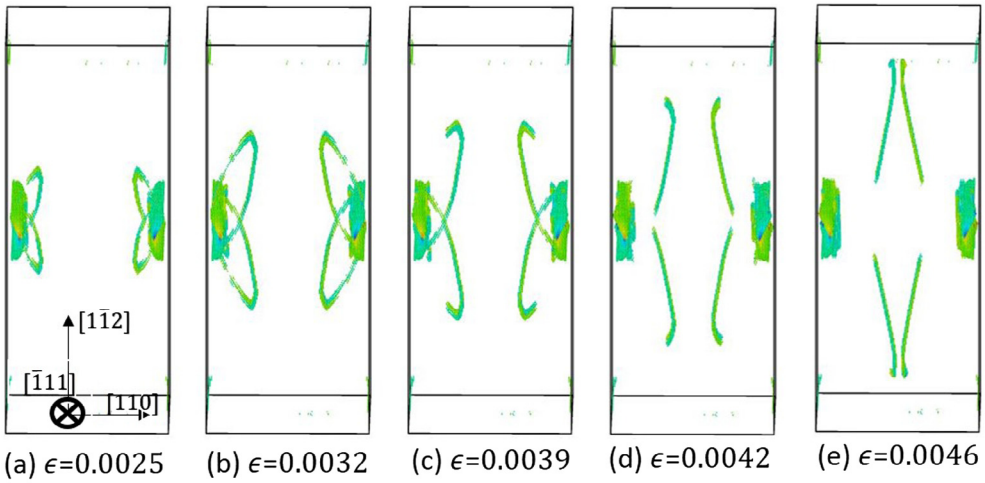

Fig. 11. Dislocations nucleation and migration in a billion-atom sample by CAC.

CAC simulation using an embedded atom method (EAM) force field [44] is performed. This simulation employs the Mishin-embedded atom method force field for single crystalline copper. In the CAC simulator, the electron charge density is calculated only on the nodes of elements. The electron charge density of those atoms within the element is interpolated from the electron charge density on the FE nodes. A single crystalline cubic f.c.c. sample in a dimension of $70\,\text{nm} \times 70\,\text{nm} \times 70\,\text{nm}$ containing $54,683,330$ atoms is discretized into $24,890$ coarse elements (the Inset pictures in Fig. 13). A constant velocity of $1\,\text{m/s}$ is applied on the two ends of the sample with the other surfaces traction free. The dislocations activities are analyzed using the dislocation extraction algorithm in OVITO [45]. Fig. 13 shows that dislocations nucleate from the free surfaces and propagate into the interior of the sample. Similar with the simulation from the CAC model using the L-J potential, dislocations also have Burges vector $\frac{1}{6}[112]$, which are consistent with those found in the full MD simulations of Cu under tension [46]. Those dislocations interact with each other, form stair-rod lock structure [45] and then a dislocation forest as those in MD simulations [4]. This preliminary simulation demonstrates that, despite the approximations introduced by the coarse mesh in CAC, the atomistic nature associated with dislocation nucleation, interactions, and the formation of sessile structures have been captured at a fraction of the cost of full MD simulations.

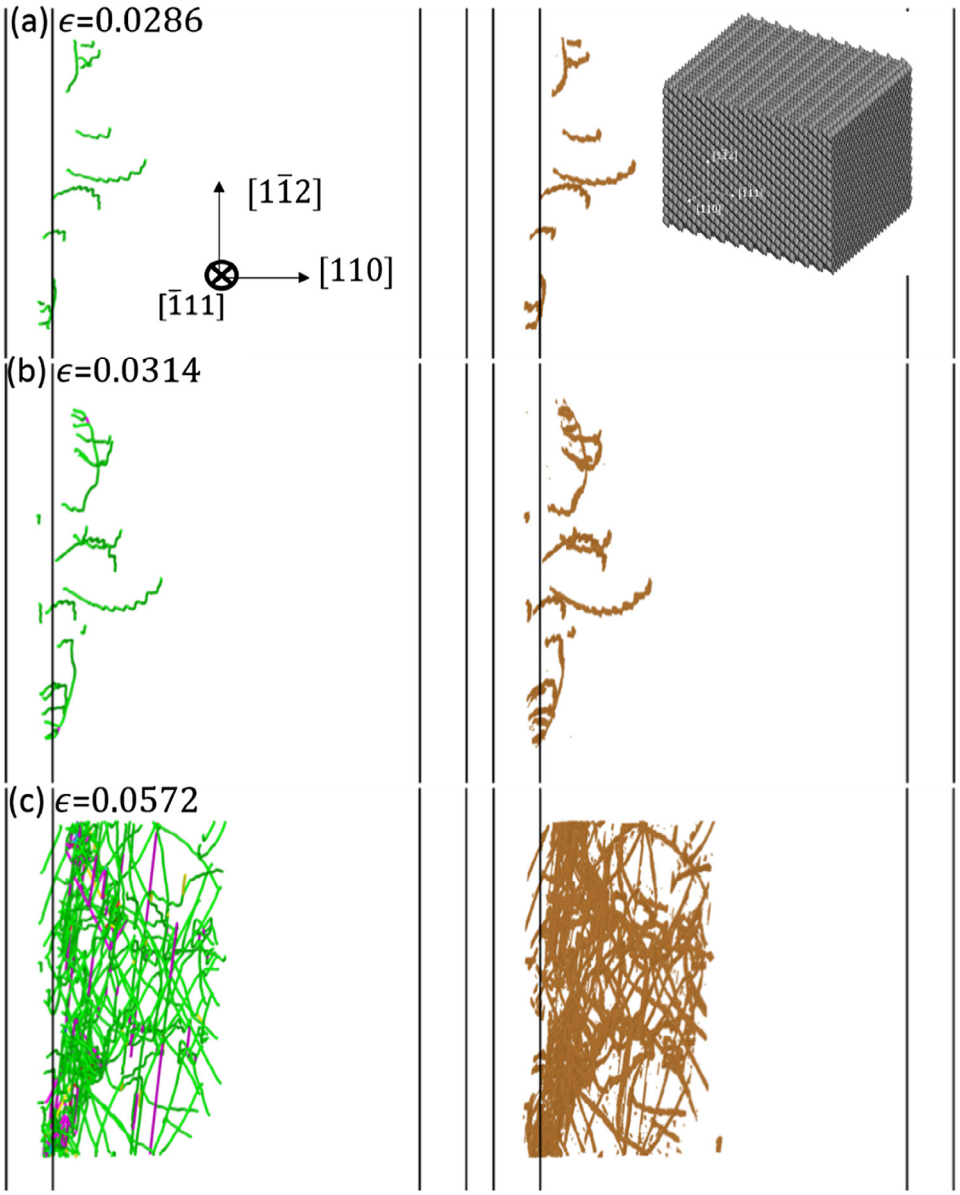

Fig. 13. CAC ($24,890$ coarse elements) simulations of dislocations nucleation and migration in a single crystalline EAM-Cu containing $54,683,330$ atoms: (a) dislocations nucleate from the free surface on one side.=; (b) the nucleated dislocations migrate into the interior of the sample; (c) the dislocation forests are formed as more and more dislocations interact with each other to form stair-rod locks [4,23]. Here the dislocation analysis is conducted using two different approaches. The dislocation structure in the left column is from the dislocation extraction algorithm in OVITO [45]. This analysis shows that the dislocations (green line) in this simulation are in Burges vector of $\frac{1}{6}[112]$ and the purple lines are the stair-rod locks. The dislocation structure in the right column is from the centrosymmetry analysis [45], in which the atoms with centrosymmetry parameter smaller than 0.1 are removed. The black lines above are boundaries of the box. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

8. Summary and discussions

In this work, a SD parallel algorithm for a multiscale simulator, CAC, is developed and implemented. Results obtained using the newly parallelized CAC simulator are directly compared with those from MD simulations. It shows that the newly developed CAC simulator can effectively reproduce dislocation nucleation, migration and the formation of Lomer-Cotrell lock formation as that in MD simulations. Furthermore, the parallel algorithm has been tested in CAC using different number of processors for different models. Using only 512 processors, CAC furnished with this new algorithm exhibits an optimal scalability in computer models which contains up to $4,809,108$ elements for $10,565,610,276$ atoms. This is beyond the reach of classical MD simulator using the same computational resource. The parallel efficiency is shown to be more than 90% and is compared with a well-established atomistic simulator, LAMMPS. For the CAC models with the atomic-scale finite element meshes which reproduce the full MD simulation results, the parallel algorithm achieves 97% efficiency of LAMMPS. To demonstrate the capability of the newly parallelized CAC simulator, dislocation activities in a large sample containing $2,307,616,753$ atoms are simulated. The maximum positive shear stress around a dislocation core in the finite-sized sample was found to gradually decay when it get closer the free surface. It approaches the analytical solution for the stress field around a dislocation core embedded in an infinite media when the core is at least $\sim 120nm$ away from the free surfaces.

It should be noted that the majority of existing concurrent multiscale method is based on domain decomposition [2,17–23], the present SD parallel algorithm can be applied to those methods and provide a general framework for parallelizing many other multiscale materials simulators. In order to use CAC to simulate more complicated phenomena such as dislocation interaction with obstacles such as voids or grain boundaries, the atomic-scale finite element mesh nearby the obstacles needs to be combined with coarse elements. The computer models with non-uniform meshes with different element sizes in the newly parallelized CAC code will be tested in our future work.

Acknowledgements

HC, JR, TP and LX acknowledge the support of NSF (Grant No. CMMI-1536925) and the Extreme Science and Engineering Discovery Environment (XSEDE allocations TG MSS170003). The work of SX was supported in part by the Elings Prize Fellowship in Science offered by the California NanoSystems Institute (CNSI) on the UC Santa Barbara campus.

References

[1] D.L. McDowell, A perspective on trends in multiscale plasticity, Int. J. Plast. 26 (9) (2010) 1280–1309.
[2] M. Wallin, W.A. Curtin, M. Ristinmaa, A. Needleman, Multi-scale plasticity modeling: coupled discrete dislocation and continuum crystal plasticity, J. Mech. Phys. Solids 56 (11) (2008) 3167–3180.
[3] V. Yamakov, D. Wolf, S.R. Phillpot, A.K. Mukherjee, H. Gleiter, Dislocation processes in the deformation of nanocrystalline aluminium by molecular-dynamics simulation, Nat. Mater. 1 (1) (2002) 45–49.
[4] M.J. Buehler, A. Hartmaier, H. Gao, M. Duchaineau, F.F. Abraham, Atomic plasticity: description and analysis of a one-billion atom simulation of ductile materials failure, Comp. Meth. Appl. Mech. Eng. 193 (48) (2004) 5257–5282.
[5] F. Roters, P. Eisenlohr, L. Hantcherli, D.D. Tjahjanto, T.R. Bieler, D. Raabe, Overview of constitutive laws, kinematics, homogenization and multiscale methods in crystal plasticity finite-element modeling: Theory, experiments, applications, Acta Mater. 58 (4) (2010) 1152–1211.
[6] R.J. Amodeo, N.M. Ghoniem, Dislocation dynamics. I. A proposed methodology for deformation micromechanics, Phys. Rev. B 41 (10) (1990) 6958.
[7] R.J. Amodeo, N.M. Ghoniem, Dislocation dynamics. II. Applications to the formation of persistent slip bands, planar arrays, and dislocation cells, Phys. Rev. B 41 (10) (1990) 6968.
[8] E. Van der Giessen, A. Needleman, Discrete dislocation plasticity: a simple planar model, Modell. Simul. Mater. Sci. Eng. 3 (5) (1995) 689.
[9] J.Q. Broughton, F.F. Abraham, N. Bernstein, E. Kaxiras, Concurrent coupling of length scales: methodology and application, Phys. Rev. B 60 (4) (1999) 2391.
[10] W. Cai, M. de Koning, V.V. Bulatov, S. Yip, Minimizing boundary reflections in coupled-domain simulations, Phys. Rev. Lett. 85 (15) (2000) 3213.
[11] M. Zhou, D.L. McDowell, Equivalent continuum for dynamically deforming atomistic particle systems, Philos. Mag. A 82 (13) (2002) 2547–2574.
[12] G.J. Wagner, W.K. Liu, Coupling of atomistic and continuum simulations using a bridging scale decomposition, J. Comput. Phys. 190 (1) (2003) 249–274.
[13] S.P. Xiao, T. Belytschko, A bridging domain method for coupling continua with molecular dynamics, Comp. Meth. Appl. Mech. Eng. 193 (17) (2004) 1645–1669.
[14] P.A. Klein, J.A. Zimmerman, Coupled atomistic–continuum simulations using arbitrary overlapping domains, J. Comput. Phys. 213 (1) (2006) 86–116.
[15] L.E. Shilkrot, W.A. Curtin, R.E. Miller, A coupled atomistic/continuum model of defects in solids, J. Mech. Phys. Solids 50 (10) (2002) 2085–2106.
[16] L.E. Shilkrot, R.E. Miller, W.A. Curtin, Coupled atomistic and discrete dislocation plasticity, Phys. Rev. Lett. 89 (2) (2002) 025501.
[17] L.E. Shilkrot, R.E. Miller, W.A. Curtin, Multiscale plasticity modeling: coupled atomistics and discrete dislocation mechanics, J. Mech. Phys. Solids 52 (4) (2004) 755–787.
[18] R.E. Miller, L.E. Shilkrot, W.A. Curtin, A coupled atomistics and discrete dislocation plasticity simulation of nanoindentation into single crystal thin films, Acta Mater. 52 (2) (2004) 271–284.
[19] M. Dewald, W.A. Curtin, Analysis and minimization of dislocation interactions with atomistic/continuum interfaces, Modell. Simul. Mater. Sci. Eng. 14 (3) (2006) 497.
[20] E. Biyikli, A.C. To, Multiresolution molecular mechanics: Implementation and efficiency, J. Comput. Phys. 328 (2017) 27–45.
[21] R.E. Miller, E.B. Tadmor, Hybrid continuum mechanics and atomistic methods for simulating materials deformation and failure, MRS Bulletin 32 (11) (2007) 920–926.
[22] R.E. Miller, E.B. Tadmor, A unified framework and performance benchmark of fourteen multiscale atomistic/continuum coupling methods, Modell. Simul. Mater. Sci. Eng. 17 (5) (2009) 053001.
[23] L. Xiong, G. Tucker, D.L. McDowell, Y. Chen, Coarse-grained atomistic simulation of dislocations, J. Mech. Phys. Solids 59 (2) (2011) 160–177.
[24] S. Xu, R. Che, L. Xiong, Y. Chen, D.L. McDowell, A quasistatic implementation of the concurrent atomistic-continuum method for FCC crystals, Int. J. Plast. 72 (2015) 91–126.
[25] S. Xu, L. Xiong, Y. Chen, D.L. McDowell, Sequential slip transfer of mixed-character dislocations across Σ3 coherent twin boundary in FCC metals: a concurrent atomistic-continuum study, npj Comput. Mater. 2 (2016) 15016.
[26] S. Xu, L. Xiong, Y. Chen, D.L. McDowell, An analysis of key characteristics of the Frank-Read source process in FCC metals, J. Mech. Phys. Solids 96 (2016) 460–476.
[27] L. Xiong, S. Xu, D.L. McDowell, Y. Chen, Concurrent atomistic–continuum simulations of dislocation–void interactions in fcc crystals, Int. J. Plast. 65 (2015) 33–42.
[28] S. Yang, L. Xiong, Q. Deng, Y. Chen, Concurrent atomistic and continuum simulation of strontium titanate, Acta Mater. 61 (1) (2013) 89–102.
[29] S. Yang, Y. Chen, Concurrent atomistic and continuum simulation of bi-crystal strontium titanate with tilt grain boundary, in: Proc. R. Soc. A, vol. 471, no. 2175 (2015, March), p. 20140758. The Royal Society.
[30] Q. Deng, Y. Chen, A coarse-grained atomistic method for 3D dynamic fracture simulation, Int. J. Multiscale Comput. Eng. 11 (3) (2013).
[31] L. Xiong, J. Rigelesaiyin, X. Chen, S. Xu, D.L. McDowell, Y. Chen, Coarse-grained elastodynamics of fast moving dislocations, Acta Mater. 104 (2016) 143–155.
[32] L. Xiong, X. Chen, N. Zhang, D.L. McDowell, Y. Chen, Prediction of phonon properties of 1D polyatomic systems using concurrent atomistic–continuum simulation, Arch. Appl. Mech 84 (2014) 1665–1675.
[33] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. 117 (1) (1995) 1–19.
[34] Y. Chen, Local stress and heat flux in atomistic systems involving three-body forces, J. Chem. Phys. 124 (5) (2006) 054113.
[35] Y. Chen, Reformulation of microscopic balance equations for multiscale materials modeling, J. Chem. Phys. 130 (13) (2009) 134706.
[36] Y. Chen, J. Lee, Atomistic formulation of a multiscale field theory for nano/micro solids, Phil. Mag. 85 (33–35) (2005) 4095–4126.
[37] F. Pavia, W.A. Curtin, Parallel algorithm for multiscale atomistic/continuum simulations using LAMMPS, Modell. Simul. Mater. Sci. Eng. 23 (5) (2015) 055002.
[38] Emre Biyikli, C. To. Albert, Multiresolution molecular mechanics: Implementation and efficiency, J. Comput. Phys. 328 (2017) 27–45.
[39] R.J. Hardy, Formulas for determining local properties in molecular-dynamics simulations: Shock waves, J. Chem. Phys. 76 (1) (1982) 622–628.
[40] http://hpcgroup.public.iastate.edu/HPC/Condo/homepage.html.
[41] V. Yamakov, D. Wolf, S.R. Phillpot, H. Gleiter, Dislocation–dislocation and dislocation–twin reactions in nanocrystalline Al by molecular dynamics simulation, Acta Mater. 51 (14) (2003) 4135–4147.

[42] S. Izumi, S. Yip, Dislocation nucleation from a sharp corner in silicon, J. Appl. Phys. 104 (3) (2008) 033513.
[43] Peter M. Anderson, John P. Hirth, Jens Lothe, Theory of Dislocations, 3., Cambridge University Press, 2016.
[44] Y. Mishin, M.J. Mehl, D.A. Papaconstantopoulos, A.F. Voter, J.D. Kress, Structural stability and lattice defects in copper: ab initio, tight-binding, and embedded-atom calculations, Phys. Rev. B 63 (22) (2001) 224106.
[45] A. Stukowski, Visualization and analysis of atomistic simulation data with OVITO–the open visualization tool, Modell. Simul. Mater. Sci. Eng. 18 (1) (2009) 015012.
[46] I. Shabib, R.E. Miller, Deformation characteristics and stress–strain response of nanotwinned copper via molecular dynamics simulation, Acta Mater. 57 (15) (2009) 4364–4373.