Materials Theory

ORIGINAL ARTICLE Open Access

Parallel-GPU-accelerated adaptive mesh refinement for three-dimensional phase-field simulation of dendritic growth during solidification of binary alloy

Shinji Sakane[1*], Tomohiro Takaki[1] and Takayuki Aoki[2]

*Correspondence:
sakane@kit.ac.jp
[1] Faculty of Mechanical Engineering, Kyoto Institute of Technology, Matsugasaki, Sakyo-ku, Kyoto 606-8585, Japan
Full list of author information is available at the end of the article

Abstract

In the phase-field simulation of dendrite growth during the solidification of an alloy, the computational cost becomes extremely high when the diffusion length is significantly larger than the curvature radius of a dendrite tip. In such cases, the adaptive mesh refinement (AMR) method is effective for improving the computational performance. In this study, we perform a three-dimensional dendrite growth phase-field simulation in which AMR is implemented via parallel computing using multiple graphics processing units (GPUs), which provide high parallel computation performance. In the parallel GPU computation, we apply dynamic load balancing to parallel computing to equalize the computational cost per GPU. The accuracy of an AMR refinement condition is confirmed through the single-GPU computations of columnar dendrite growth during the directional solidification of a binary alloy. Next, we evaluate the efficiency of dynamic load balancing by performing multiple-GPU parallel computations for three different directional solidification simulations using a moving frame algorithm. Finally, weak scaling tests are performed to confirm the parallel efficiency of the developed code.

Keywords: Phase-field method, Dendrite growth, Solidification microstructure, Graphics processing unit, Adaptive mesh refinement, Parallel computing

Introduction

A solidification microstructure is generally composed of equiaxed and columnar structures, which are formed by the competitive growth of a massive number of dendrites. As the solidification microstructure determines the macroscopic properties of a cast product, it is important to accurately predict the formation process of the solidification microstructure considering the competitive growth (Dantzig and Rappaz 2009; Kurz et al. 2018, 2020).

There are a few numerical simulation methods that can express dendrite growth, such as the cellular automaton method (Brown et al. 1994; Wei et al. 2012; Chen et al. 2015), front tracking method (Juric and Tryggvason 1996; Al-Rawahi and Tryggvason

2002, 2004), and phase-field (PF) method (Gránásy et al. 2013; Takaki 2014; Plapp 2016). Among these, the PF method is the most powerful because it does not require the explicit tracking of an interface and accurately expresses the free-boundary problem through the thin-interface limit. However, the PF method is a diffuse interface model, which requires a finer spatial resolution compared to other simulation methods, resulting in higher calculation costs. A quantitative solidification model (Ohno 2020) can be utilized to perform quantitative simulations using an interface width larger than the physical interface. However, even in a quantitative PF model, it is necessary to use a numerical grid that is several times smaller than the curvature radius of a dendrite tip. Therefore, the PF simulations of dendrite growth were limited to the two-dimensional (2D) simulations and three-dimensional (3D) simulations of single dendrite growth until the early 2000s (Kobayashi 1994; Karma and Rappel 1998; Karma et al. 2000).

A graphics processing unit (GPU) was first used for PF computation in the early 2000s. Currently, GPUs are commonly used in the PF simulations of dendritic solidification owing to their high parallel computing performance (Yamanaka et al. 2011; Ohno 2012; Tourret and Karma 2015; Clarke et al. 2017; Song et al. 2018). GPU computing is quite efficient for implementing discretization methods with simple calculation algorithms, such as finite difference or finite volume methods. In addition, the PF computation for dendrite growth is easily accelerated by GPU computing because it requires a large number of floating-point calculations per grid point. Large-scale simulations with multiple GPUs (Shimokawabe et al. 2011; Sakane et al. 2015) were performed for the competitive growth of multiple 3D columnar dendrites during directional solidification (Takaki et al. 2016a, b, 2018). Although GPU computing is quite powerful for dendrite growth PF simulations, it remains limited in scale for the simulations with multiple dendrites when the diffusion length is considerably larger than the dendrite tip radius (Bellón et al. 2021).

The adaptive mesh refinement (AMR) method is used to improve the efficiency of PF computation for dendritic solidification (Provatas et al. 1999; Lan et al. 2003; Takaki et al. 2005; Guo et al. 2014). In recent years, multiple dendrite growth in a 3D system has been simulated by the PF method using parallel AMR computing on a central processing unit (CPU) cluster (Guo and Xiong 2015; Gong et al. 2018; Greenwood et al. 2018; Wang et al. 2021). Recently, the applications of parallel AMR have been extended to multiphysics problems in dendrite growth, including thermal diffusion (Zhang et al. 2020), melt convection (Zhang et al. 2021a), and gas porosity (Zhang et al. 2021b). Although a few physical simulations (Schive et al. 2018; Watanabe and Aoki 2021) implemented AMR using parallel computing with multiple GPUs, PF simulations for dendrite growth have not yet applied AMR using GPUs. AMR is extremely effective in resolving problems such as a wide primary arm spacing compared to the curvature radius of a dendrite tip. In these conditions, the further acceleration of PF simulations via AMR is required to treat a large number of dendrites.

In this study, the AMR method is implemented using multiple-GPU parallel computing to increase the computational scale and efficiency of 3D PF dendrite growth simulations. In multiple-GPU computing, if a fixed subdomain is assigned to one GPU, the number of grid points for each GPU differs depending on the state of simulations, thereby reducing the parallel computation performance. Thus, dynamic load balancing is introduced in domain subdivision to equalize the number of grid points for each

GPU. The developed multiple-GPU parallel computing method for AMR is evaluated by simulating columnar dendrite growth during the directional solidification of a binary alloy. First, an AMR condition is evaluated by performing single-GPU computation. Next, dynamic load balancing is evaluated through multiple-GPU parallel computation. Finally, the parallel computation performance is investigated by conducting weak scaling tests by increasing the number of GPUs.

Method

PF model

A quantitative PF model for the directional solidification of dilute binary alloys (Ohno and Matsuura 2009) is employed for the simulations. In this model, the temperature field is assumed to be $T = T_r + Gz - Rt$ on the basis of the frozen temperature approximation (Diepers et al. 2002). Here, T_r is the reference temperature, G is the temperature gradient, and R is the cooling rate. The morphological evolution of dendrites is expressed by solving the PF equation. The solid and liquid phases are expressed by the PF variable, ϕ, where $\phi = +1$ and $\phi = -1$ in the solid and liquid phases, respectively, and ϕ varies smoothly at the solid–liquid interface. Nonlinear preconditioning, $\phi = \tanh(\psi/\sqrt{2})$, is applied to the PF variable (Gong et al. 2018; Glasner 2001) to perform stable computations using a large grid size. The solute concentration is treated as dimensionless supersaturation, i.e., $u = (C_l - C_l^e)/(C_l^e - C_s^e)$, where C_l and C_s denote the solute concentrations and C_l^e and C_s^e denote the equilibrium solute concentrations in the liquid and solid phases, respectively. The solute concentration, C, is expressed as $C = (1 - \tanh(\psi/\sqrt{2}))C_l/2 + (1 + \tanh(\psi/\sqrt{2}))C_s/2$. It is assumed that the relation for the partition coefficient, $k = C_s^e/C_l^e = C_s/C_l$, is satisfied on the basis of the dilute solution approximation (Kim et al. 1999).

The time evolution equations for the preconditioned PF variable, ψ, and dimensionless solute concentration, u, are expressed as

$$\begin{aligned}\tau\left[1-(1-k)u'\right]\cdot\frac{\partial\psi}{\partial t} &= W^2\nabla^2\psi - \sqrt{2}W^2\tanh\left(\frac{\psi}{\sqrt{2}}\right)|\nabla\psi|^2 \\ &+ \sum_{r=x,y,z}\frac{\partial}{\partial r}\left[W\frac{\partial W}{\partial\psi_r}|\nabla\psi|^2\right] + \sqrt{2}\tanh\left(\frac{\psi}{\sqrt{2}}\right) - \sqrt{2}\lambda^*\left(1-\tanh^2\left(\frac{\psi}{\sqrt{2}}\right)\right)(u+u') \\ &+ 2W\nabla W\cdot\nabla\psi - \sqrt{2}W\tanh\left(\frac{\psi}{\sqrt{2}}\right)|\nabla\psi|^2\sum_{r=x,y,z}\left[\frac{\partial W}{\partial\psi_r}\cdot\frac{\partial\psi}{\partial r}\right],\end{aligned} \tag{1}$$

$$\begin{aligned}\frac{1}{2}\left[1+k-(1-k)\tanh\left(\frac{\psi}{\sqrt{2}}\right)\right]\cdot\frac{\partial u}{\partial t} &= \nabla(D_l q(\psi)\nabla u - \mathbf{J}_{AT}) \\ &+ \frac{1}{2\sqrt{2}}[1+(1-k)u]\left(1-\tanh^2\left(\frac{\psi}{\sqrt{2}}\right)\right)\cdot\frac{\partial\psi}{\partial t} - \nabla\cdot\mathbf{J},\end{aligned} \tag{2}$$

where $\tau = \tau_0 a_s(\mathbf{n})^2$ is the PF relaxation time and $W = W_0 a_s(\mathbf{n})$ is the interface width. $a_s(\mathbf{n})$ is a crystal anisotropy function:

$$a_s(\mathbf{n}) = (1-3\varepsilon_4)\left[1 + \frac{4\varepsilon_4}{1-3\varepsilon_4}\frac{\sum_{r=x,y,z}(\partial\psi/\partial r)^4}{|\nabla\psi|^4}\right], \tag{3}$$

where ε_4 is the anisotropic strength. u' in Eq. (1) represents directional solidification, and it is expressed as $u' = -(Gz - Rt)/(m(C_l^e - C_s^e))$. Here, m is the liquidus slope. D_l and D_s are the diffusion coefficients of the solute in the liquid and solid phases, respectively, and $q(\psi)$ is the interpolation function, $q(\psi) = [kD_s + D_l + (kD_s - D_l)\tanh(\psi/\sqrt{2})]/(2D_l)$. $\nabla \cdot \mathbf{J}$ is the noise term for generating higher-order dendrite arms. $\mathbf{J}_{\mathrm{AT}}$ is an antitrapping current term for correcting the solute flux at the interface:

$$\mathbf{J}_{AT} = -\frac{1}{4}\left(1 - \frac{kD_s}{D_l}\right) W_0[1 + (1-k)u]\left(1 - \tanh^2\left(\frac{\psi}{\sqrt{2}}\right)\right) \cdot \frac{\partial \psi}{\partial t}\frac{\nabla \psi}{|\nabla \psi|}. \tag{4}$$

The time and spatial derivative terms in Eqs. (1) and (2) are calculated using the finite difference method. The first-order forward difference scheme is used for the time derivative term. The isotropic difference scheme (Shimokawabe et al. 2011) and second-order central difference scheme are used for the spatial second-order derivative terms in Eqs. (1) and (2), respectively.

AMR method

An adaptive mesh is generated on the basis of the block-structured AMR method (Schive et al. 2018). In this method, as shown on the left in Fig. 1, the entire computational domain is first divided into the coarsest cubic blocks. The refinement level of these blocks is defined as 0, and a level n block is obtained by recursively dividing a level 0 block into eight blocks n times based on the octree structure. In Fig. 1, for example, a level 2 block is the finest block. Here, the difference in the refinement level between two adjacent blocks is constrained to be no greater than 1. As shown by the red frame in Fig. 1, each block is divided into N^3 grids, where the finite difference computations of Eqs. (1) and (2) are performed. According to the maximum refinement level, $n_{\max}$, and the minimum grid size, $\Delta x_{\min}$, the size of the grids belonging to a level n block is $\Delta x_n = 2^{(n_{\max}-n)}\Delta x_{\min}$ and the size of a level 0 block is $X_0 = N\Delta x_0 = 2^{(n_{\max}-n)}N\Delta x_{\min}$. Here, the lengths of each side of the computational domain (L_x, L_y, and L_z) are set as integral multiples of X_0. The block information, such as the refinement level, position, and adjacent blocks, is stored on the GPU to reduce the data communication between the GPU and CPU. The information is updated when the refinement and coarsening of the mesh are performed on the GPU. At the beginning of the simulations, fixed-length memory arrays are allocated for storing physical data in each GPU. The length of the memory array is set to be as long as possible within the limits of the GPU memory. Each GPU must contain all the information in the allocated subdomain in the GPU memory arrays. When the GPU memory is insufficient to contain the subdomain information, the simulation is paused. The simulation can be continued by further dividing the subdomains and increasing the number of GPUs used for the simulation. For each block, the physical data of $(N+2)^3$ points, including the halo region, are lined up on the allocated memory array. At the beginning of each time step, the physical data of the grid points located on the boundary of a block are transferred to the halo region of the adjacent block.

In the following evaluations of the implemented dynamic load balancing, the moving frame algorithm (MFA) (Diepers et al. 2002; Boettinger and Warren 1999) is used in the directional solidification simulations to intentionally shift the distribution of dendrites in the computational domain. Figure 2 shows the images of the columnar dendrite growth

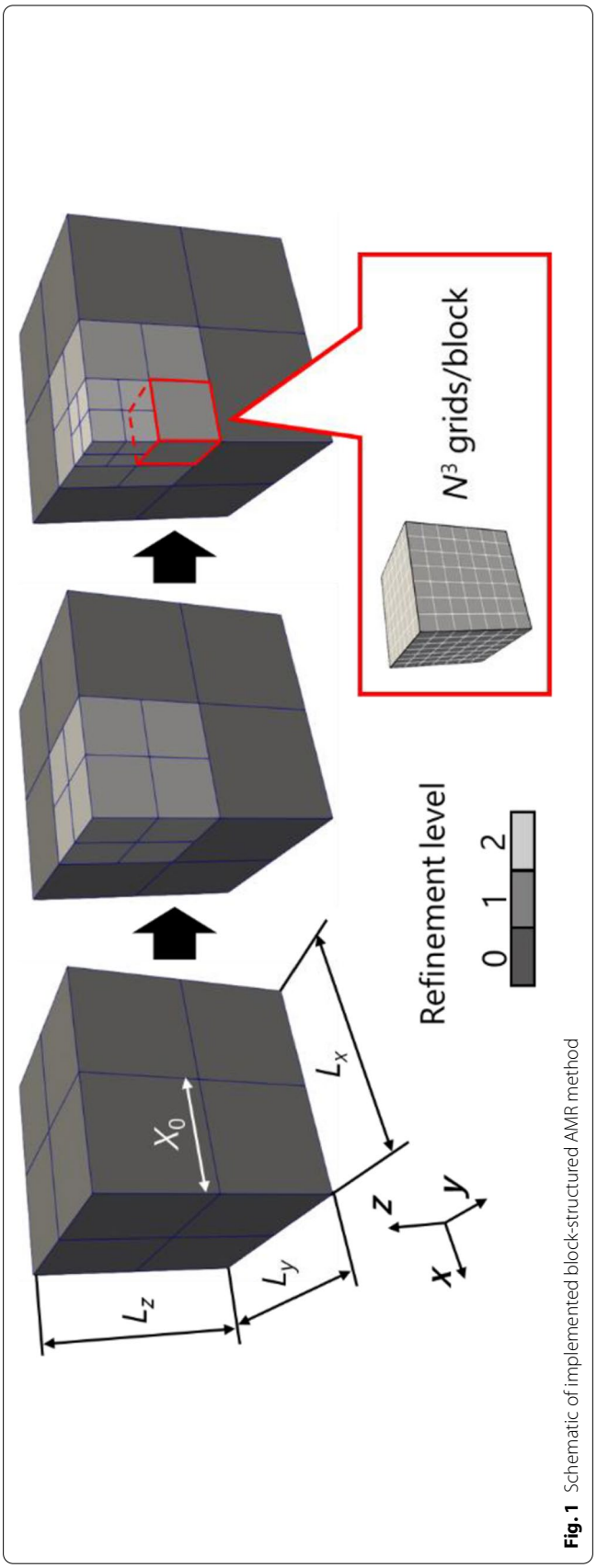

Fig. 1 Schematic of implemented block-structured AMR method

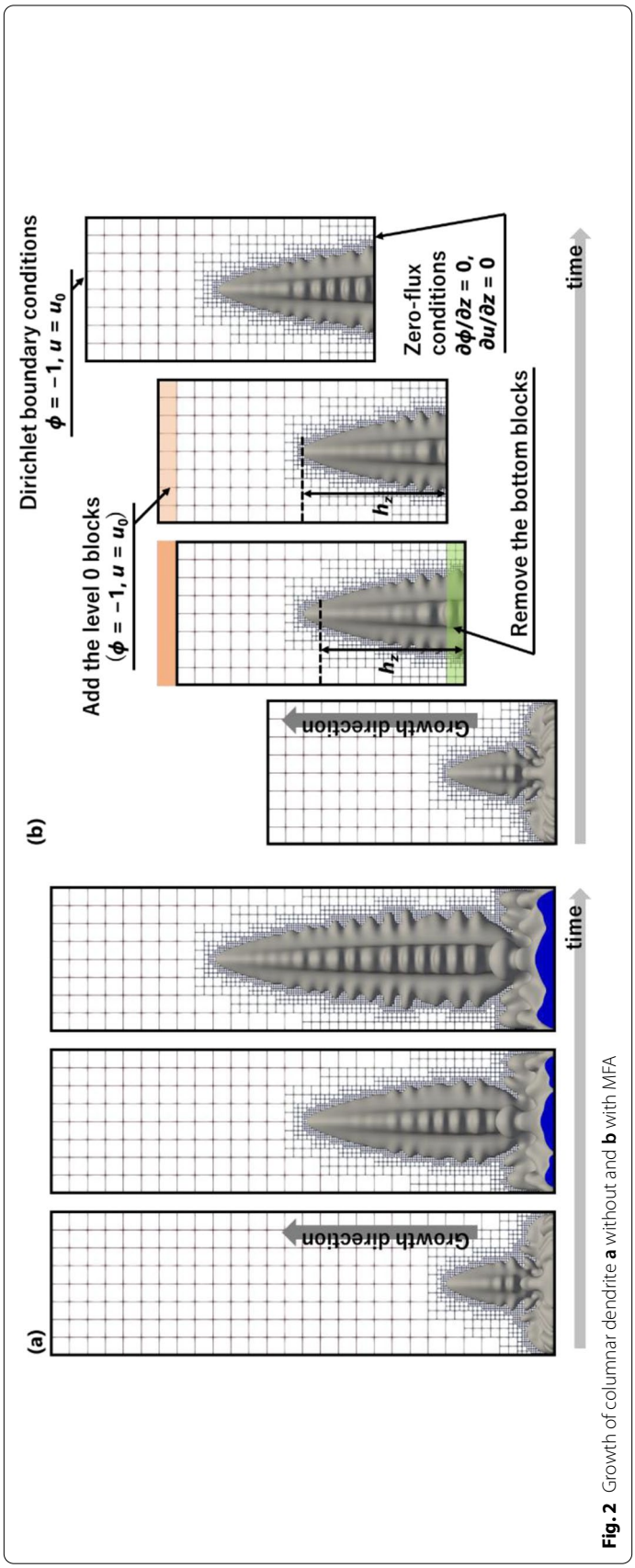

Fig. 2 Growth of columnar dendrite **a** without and **b** with MFA

during directional solidification with and without the MFA. Figure 2a shows a typical directional solidification simulation without the MFA, where the dendrite tip moves upward in the computational domain. In simulations with the MFA, the dendrite tip position is maintained at a position in the growth direction. The procedure of removing the bottom region (green region) and adding the top region (orange region) is repeated to maintain the dendrite length at h_z or lower. When a uniform grid is used, the domain is typically moved by one grid. In contrast, in the present AMR, the domain is moved by the level 0 block size, X_0, as shown in Fig. 2b. The Dirichlet boundary condition ($\phi=-1$, $u=u_0$) and zero-flux boundary condition are applied to the top and bottom boundaries, respectively.

Multiple-GPU parallelization

Parallel computation is implemented using multiple GPUs to increase the computational scale of the 3D-AMR-PF simulations. At the beginning of the simulation, the computational domain is evenly divided into M_x, M_y, and M_z subdomains in the x, y, and z directions, respectively, and one GPU is assigned to each subdomain. As the number of grid points is different for each subdomain owing to the application of AMR, the subdomain boundaries dynamically slide based on the slice-grid method (Tsuzuki and Aoki 2016) such that the number of grid points for each GPU becomes the same. We select the slice-grid method because of its simplicity; however, there are more complicated schemes (Watanabe et al. 2020). Here, the slide distances of the subdomain boundaries are set as integral multiples of X_0. Figure 3 shows an example of how the subdomain boundary positions are determined in the dynamic load balancing, when $M_x=M_y=M_z=4$. First, the position of the three subdomain boundaries in the x direction is determined such that the ratio of the number of grid points in the two regions separated by the i^{th} boundary becomes approximately $i{:}M_x-i$. Next, the same procedure is repeated for the y direction, in the four regions sandwiched by two boundaries in the x direction, as shown in Fig. 3c. Finally, the boundary positions in the z direction are determined in the same manner for the 16 regions sandwiched between the x-direction and y-direction boundaries, as shown in Fig. 3d. In the multiple-GPU computation, it is necessary to share the physical data of the blocks adjacent to subdomain boundaries. The sharing procedure is performed as follows. At the beginning of each time step, the physical data of each block adjacent to a subdomain boundary are transferred from a GPU to the host CPU. Then, the transferred data are individually sorted according to the adjacent subdomain boundary, after which they are transferred to the CPU assigned to the corresponding adjacent subdomain through a message-passing interface. Finally, the received data are transferred from the adjacent subdomain CPU to the corresponding GPU. The computational code is written in CUDA C, and OpenMPI is used for internode communication. The subsequent simulations are performed using the TSUBAME3.0 GPU supercomputer (Tokyo Institute of Technology). TSUBAME3.0 consists of 540 computing nodes with 2 CPUs (Intel Xeon E5-2680 V4 Processor) and 4 GPUs (NVIDIA Tesla P100) per node.

Simulations

The accuracy of the AMR refinement condition used in this study and the performance of the dynamic load balancing are evaluated using single-GPU and multiple-GPU computations, respectively. Finally, weak scaling tests are performed to evaluate the parallel

Fig. 3 Block decomposition of computational domain using slice-grid method: **a** surface of decomposed entire domain, **b** *x*-direction boundaries, **c** *y*-direction boundaries between 2nd and 3rd *x*-direction boundaries, and **d** *z*-direction boundaries between 2nd and 3rd *x*-direction and *y*-direction boundaries

computation performance. In the simulations, Fe-5.4wt.%Si is used as the sample material, and its material properties are shown in Table 1. The simulation conditions are also listed in Table 1.

AMR refinement condition

The AMR refinement condition is set as that the blocks with the highest level are arranged to cover only the interface region ($-0.8<\phi<0.8$) and the difference between the levels of adjacent blocks is 1 or lower. The accuracy of the refinement condition and acceleration due to AMR are evaluated by performing the single-GPU computations of columnar dendrite growth during directional solidification. The maximum refinement level is set as $n_{\max}=0$, 1, 2, 3, 4, and 5, where $n_{\max}=0$ implies that the entire domain is divided into the finest mesh, which is a uniform mesh. The number of grids per block is set as $N=8$. The AMR operation, i.e., the refinement and coarsening of the mesh, is conducted every 100 steps. The computational domain size is set as $L_x\times L_y\times L_z=0.19 2\times 0.192\times 0.384\,\mathrm{mm}^3$ ($256\Delta x_{\min}\times 256\Delta x_{\min}\times 512\Delta x_{\min}$; $\Delta x_{\min}=0.75\,\mu\mathrm{m}$). The zero-flux boundary condition is imposed on all boundaries. At the beginning of the simulation, the dimensionless supersaturation at the bottom of the domain is set as $u_0=-0.1$, and a spherical solid with a radius of $6\Delta x_{\min}$ is placed in the corner of the domain. For comparison, simulations are performed using a uniform mesh with $\Delta x=\Delta x_{\min}$.

Figure 4a shows the result at $t=2.0\times 10^5\Delta t$ in the simulation with $n_{\max}=3$, where the solute concentration on the *xy*, *yz*, and *zx* planes passing through the origin is indicated by color contours, the outlines of the blocks on the planes are shown by a white line, and the solid–liquid interface morphology is shown as a contour plane with $\phi=0$. The fine mesh is adaptively arranged only around the interface region. Figure 4b compares the solid–liquid interface morphologies on the *yz* plane at $t=2.0\times 10^5\Delta t$ (green line for $n_{\max}=3$, red line for $n_{\max}=5$, and black line for uniform mesh). All the morphologies agree, and sufficient computational accuracy can be achieved using the simple refinement conditions for $n_{\max}=3$ and 5. Figure 4c shows the plot of the execution time up to $2\times 10^5\Delta t$ vs. $n_{\max}$. The black dashed line shows the execution time for the uniform mesh. The execution time decreases as $n_{\max}$ increases. At $n_{\max}\geq 1$, the execution time for AMR is less than that for the uniform mesh. In addition, the execution time converges

Table 1 Material properties of Fe-5.4wt.%Si and simulation conditions

Quality	Symbol	Value	Reference
Diffusion coefficient in liquid	D_l	$4.40\times 10^{-9}\,\mathrm{m}^2/\mathrm{s}$	(Calderon et al. 1971)
Diffusion coefficient in solid	D_s	$3.44\times 10^{-11}\,\mathrm{m}^2/\mathrm{s}$	(Borg and Lai 1970)
Partition coefficient	k	0.79	(Cui and Jung 2017)
Anisotropic strength	ε_4	0.02	
Gibbs–Thomson constant (Fe)	Γ	2.88×10^{-7} Km	(Karrasch 2016)
Liquidus slope	m	−884 K/at.frac.	(Cui and Jung 2017)
Cooling rate	R	5 K/min	
Temperature gradient	G	3 K/mm	
Minimum grid spacing	$\Delta x_{\min}$	0.75 μm	
Interface width	W_0	1.25 μm	
Time increment	Δt	1.826×10^{-5} s	

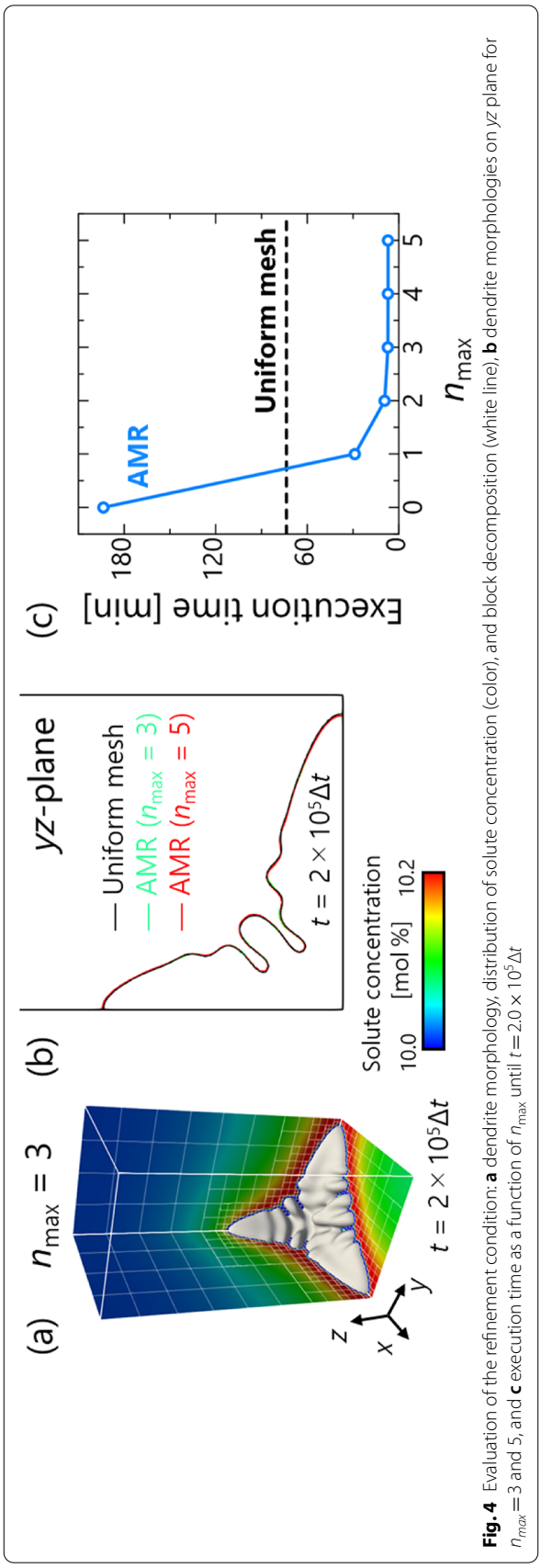

Fig. 4 Evaluation of the refinement condition: **a** dendrite morphology, distribution of solute concentration (color), and block decomposition (white line), **b** dendrite morphologies on *yz* plane for $n_{max} = 3$ and 5, and **c** execution time as a function of n_{max} until $t = 2.0 \times 10^5 \Delta t$

to an almost constant value at $n_{\max} \geq 3$. At $n_{\max} = 0$, the execution time for AMR is 2.6 times larger than that for the uniform mesh, even though the number of grid points is the same for both cases. This is because the computation with AMR requires data transfer to the halo region of each block at every time step. At $n_{\max} \geq 3$, the computation with AMR is approximately 10.3 times faster than that with the uniform mesh because the number of grid points is significantly reduced. At the initial stage of solidification, in which the solid fraction is extremely small, the acceleration ratio is approximately 50. As dendrites grow, the number of grids gradually increases and the acceleration due to AMR gradually decreases.

To compare the computational performance between GPUs and CPUs, we performed the simulation shown in Fig. 4a with the uniform mesh using a single CPU (Intel Xeon E5-2680 V4 processor) core on TSUBAME3.0. The execution time was approximately 12,700 min. Assuming that the ideal execution time with all cores of a single CPU is the execution time with a single CPU core divided by the number of cores per CPU, the ideal execution time was 912 min (as the Xeon E5-2680 V4 processor has 14 CPU cores). Compared to the ideal execution time for the single CPU, the GPU computation with the uniform mesh was 10 times faster, while the GPU computations with the adaptive mesh ($n_{\max} \geq 3$) were 100 times faster.

Dynamic load balancing

Directional solidification simulations are performed using multiple GPUs to evaluate the performance of dynamic load balancing. For this task, we change the heterogeneity of the density of the grid points in the computational domain by changing the maintained dendrite length in the computational domain with the MFA. The computational domain is a cuboid with dimensions of $L_x \times L_y \times L_z = 0.768 \times 0.768 \times 1.536\,\mathrm{mm}^3$ ($1024\Delta x_{\min} \times 1024\Delta x_{\min} \times 2048\Delta x_{\min}$; $\Delta x_{\min} = 0.75\,\mu\mathrm{m}$). At the beginning of the simulations, the supersaturation at the bottom of the domain is set as $u_0 = -0.2$, and a planar solid phase with a thickness of $6\Delta x_{\min}$ is placed at the bottom of the domain. We perform three simulations with different dendrite placements (Cases 1, 2, and 3) in the domain to evaluate the effect of dynamic load balancing. Case 1 uses a static domain without the MFA. Cases 2 and 3 use the MFA with the maximum dendrite tip position at $h_z = 0.4L_z$ and $h_z = 0.2L_z$, respectively. In Case 1, the grid points are uniformly distributed throughout the computational domain. In Case 2, the grid points are concentrated in the lower half of the domain. Finally, in Case 3, the grid points are concentrated near the bottom of the domain. Thus, the heterogeneity of the density of the grid points gradually increases in the order of Case 1, Case 2, and Case 3. Sixteen GPUs are used for the parallel computing, and the entire domain is divided into $2 \times 4 \times 2$ subdomains in the x, y, and z directions. For comparison, the simulation with the uniform mesh ($\Delta x = \Delta x_{\min}$) is performed in the same parallel computing condition with the 2D division ($1 \times 4 \times 4$ subdomains) (Sakane et al. 2015).

Figure 5 shows the solid–liquid interface morphologies and solute concentration on the xy, yz, and zx planes at different time steps for Cases 1–3. In all simulations, a typical growth procedure during directional solidification is observed: initial planar interface (initial state), perturbation ($t = 1.0 \times 10^6 \Delta t$), competitive growth of cells ($t = 1.8 \times 10^6 \Delta t$), and competitive growth of dendrites ($t \geq 2.3 \times 10^6 \Delta t$). In

Case 1 (Fig. 5a), the tip of the columnar dendrite reaches the top of the domain at $t = 2.49 \times 10^6 \Delta t$. At this time, the memory required for the computation reaches the maximum GPU memory owing to the increase in the number of grid points. In Case 2 (Fig. 5b), the computational domain starts moving from $t = 1.89 \times 10^6 \Delta t$, and then, the dendrite tip positions remain lower than h_z. At $t = 3.0 \times 10^6 \Delta t$, dendrites A and B become particularly large and other dendrites start to be eliminated. In Case 3 (Fig. 5c), the domain starts moving from $t = 1.65 \times 10^6 \Delta t$, and after $t = 2.3 \times 10^6 \Delta t$, mainly dendrites A and B remain.

Figure 6 shows the changes in subdomain decompositions for the dendrite morphologies at different time steps for Cases 1–3 (shown in Fig. 5). In all cases, at the initial state, the z-direction subdomain boundaries are located below because the grid points are concentrated in the lower part of the domain. At $t = 1.0 \times 10^6 \Delta t$, the boundaries move upward but remain in the lower part of the domain. At $t = 1.8 \times 10^6 \Delta t$ for Cases 1 and 2, the subdomain boundaries move further upward owing to the growth of numerous cells. Moreover, the domain is divided almost evenly in the x and y directions because there is almost no shift in the distribution of grid points along these

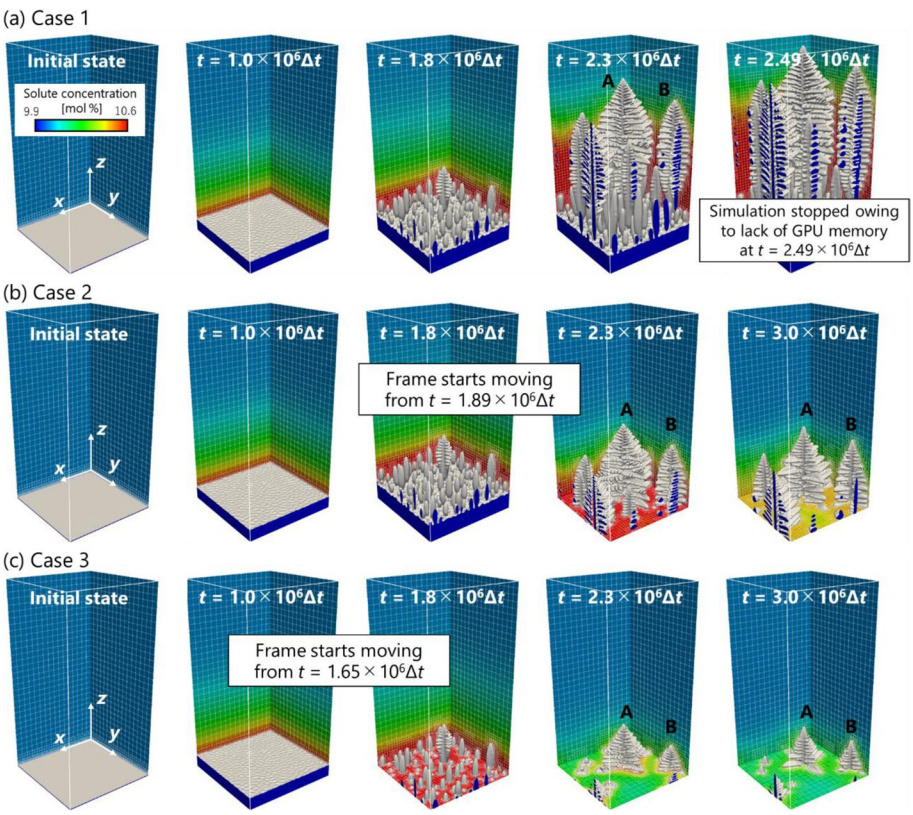

Fig. 5 Time evolutions of dendrite morphologies and solute concentration distributions during the simulations of the directional solidification from a planar solid seed, using parallel computation with 16 GPUs. In the simulations, **a** a static domain (Case 1) and domains that move in the z-direction to maintain the dendrite tip position at the z-direction positions **b** $h_z = 0.4L_z$ (Case 2) and **c** $h_z = 0.2L_z$ (Case 3) using MFA were used. The dendrite morphologies are indicated by the contours of $\varphi = 0$. The distributions of solute concentration and AMR block decompositions on the xy, yz, and zx planes are indicated by the colored contours and white lines

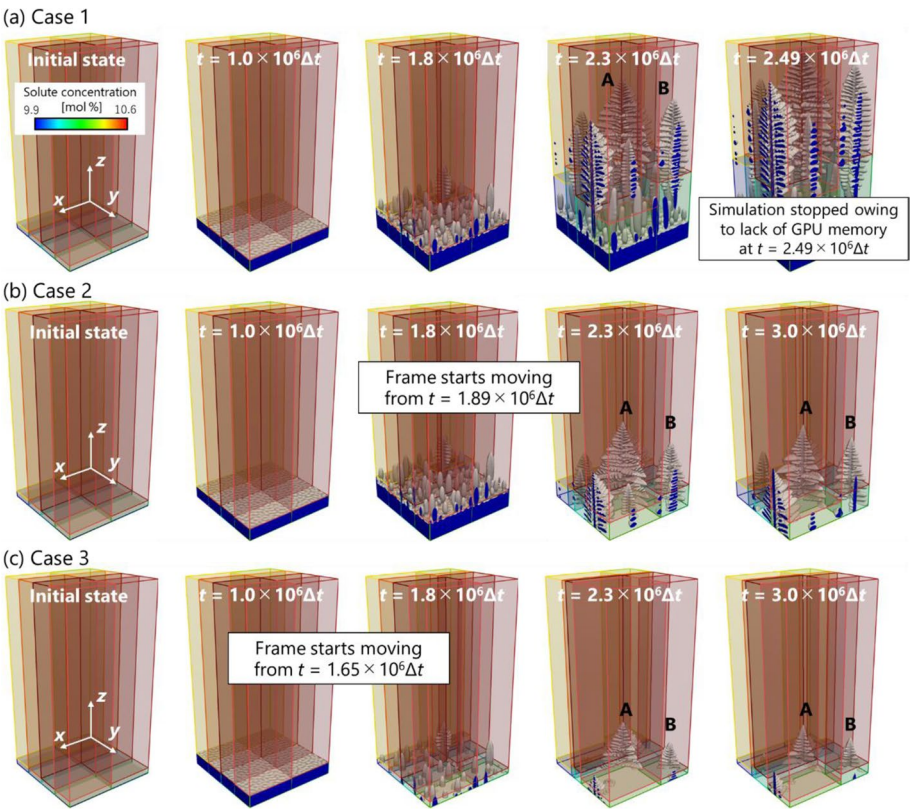

Fig. 6 Domain decomposition considering the dynamic load balancing for parallel computation using 16 GPUs during the directional solidification simulations shown in Fig. 5. In the simulations, the entire domain is divided into $2 \times 4 \times 2$ subdomains in the x, y, and z directions. The outline of each subdomain is indicated by a different color

directions. In Case 1 (Fig. 6a), at $t = 2.49 \times 10^6 \Delta t$, the ratio of the maximum and average number of grid points assigned to 16 GPUs is approximately 1.30.

In Case 2 (Fig. 6b), after $t = 2.3 \times 10^6 \Delta t$, the grid points are shifted around dendrites A and B. Accordingly, the x-direction subdomain boundary is located between dendrites A and B. In the y direction, the small subdomain (2nd from the back) is formed around the largest dendrite, i.e., dendrite A. In the z direction, the subdomain boundaries are located at a lower point compared to Case 1. The ratio of the maximum and average number of grid points assigned to 16 GPUs at $t = 3.0 \times 10^6 \Delta t$ is approximately 1.67.

In Case 3 (Fig. 6c), almost all the grid points are concentrated around dendrites A and B after $t = 2.3 \times 10^6 \Delta t$. Therefore, the smallest subdomain in the y direction is thinner than that in Case 2, and the z-direction subdomain boundary is located at a lower position than that in Case 2. At $t = 3.0 \times 10^6 \Delta t$, the ratio of the maximum and average number of grid points assigned to 16 GPUs is approximately 1.84.

In the above three cases, the ratio of the maximum and average number of grid points assigned to each GPU is less than two. This value is reasonable considering the remarkable heterogeneity of the grid points in Cases 2 and 3 and the use of the relatively simple slice-grid method. Thus, it can be concluded that appropriate domain subdivision using dynamic load balancing is conducted.

Figure 7a and b show the change in the total number of active grid points and execution time per 2000 steps, respectively, for Cases 1–3. Grid points are considered active if they are included in the AMR block placed in the computational domain. The vertical axis in Fig. 7a represents the ratio of the total number of active grid points for Cases 1–3 to the number of grid points for the uniform mesh ($= 1024 \times 1024 \times 2048$), and the vertical axis in Fig. 7b shows the execution time per 2000 steps in the simulation. In Fig. 7b, for comparison, the results obtained using the uniform mesh are indicated by the black dashed line. As shown in Fig. 7a, the number of grid points is almost constant and less than 5% of that in the uniform mesh until $t = 1.2 \times 10^6 \Delta t$, when the primary arms start growing from the flat interface. After around $t = 1.5 \times 10^6 \Delta t$, the number of grid points and execution time increase monotonically as the primary and secondary arms grow. In Case 1, the computation stops owing to the lack of GPU memory at $t = 2.49 \times 10^6 \Delta t$. In Case 2, the computational domain starts moving from $t = 1.89 \times 10^6 \Delta t$. Thereafter, the number of grid points remains at approximately 10% of that in the uniform mesh, and the execution time is approximately half of that for the uniform mesh. The fluctuations in the curves for cases 2 and 3 in Fig. 7 are caused by removing the bottom blocks in the MFA. In Case 3, the domain starts moving from $t = 1.65 \times 10^6 \Delta t$. Then, the number of grid points converges to approximately 3% of that in the uniform mesh, and the execution is approximately six times faster.

According to Fig. 7, the execution time of the adaptive mesh (case 1) and that of the uniform mesh are equal at $t = 2.2 \times 10^6 \Delta t$. Within the same time, the number of active grid points of the adaptive mesh (case 1) is 21% of that of the uniform mesh. Thus, the execution time per active grid point for the adaptive mesh is approximately five times larger than the execution time per grid point for the uniform mesh, owing to AMR and dynamic load balancing in the multiple-GPU computation. Nevertheless, the acceleration of the computation by two times (Case 2) and six times (Case 3) due to AMR exhibiting a major advantage, particularly in extremely large-scale or long-duration simulations.

Parallel computation performance

Weak scaling tests are performed to evaluate the parallel computation performance of the developed method. In the evaluations, the computational domain is set as $L_x \times L_y \times L_z = 1024P\Delta x_{\min} \times 1024Q\Delta x_{\min} \times 2048\Delta x_{\min}$, where P and Q are integers. A total of $16PQ$ GPUs are used for the evaluations. The computational conditions for ϕ and u are created by periodically arranging the results at $t = 3.0 \times 10^6 \Delta t$ in Fig. 5b. An example for 256 GPUs ($P = Q = 8$) is shown in Fig. 8. In this example, the computational conditions shown in Fig. 8a are periodically repeated in the x and y directions, as shown in Fig. 8b. The execution times for 2000 time steps from $t = 3.0 \times 10^6 \Delta t$ are measured using 16, 32, 64, 128, and 256 GPUs. During the evaluations, the refinement and coarsening of the mesh were conducted 20 times in total. It was also judged 20 times whether it was necessary to move the subdomain boundaries for dynamic load balancing; however, no boundary movement occurred during the evaluations. Because the movement of the subdomain boundaries was conducted only slightly less than 100 times throughout the simulations, the execution time for movement of subdomain boundaries is negligible in the parallel performance evaluations.

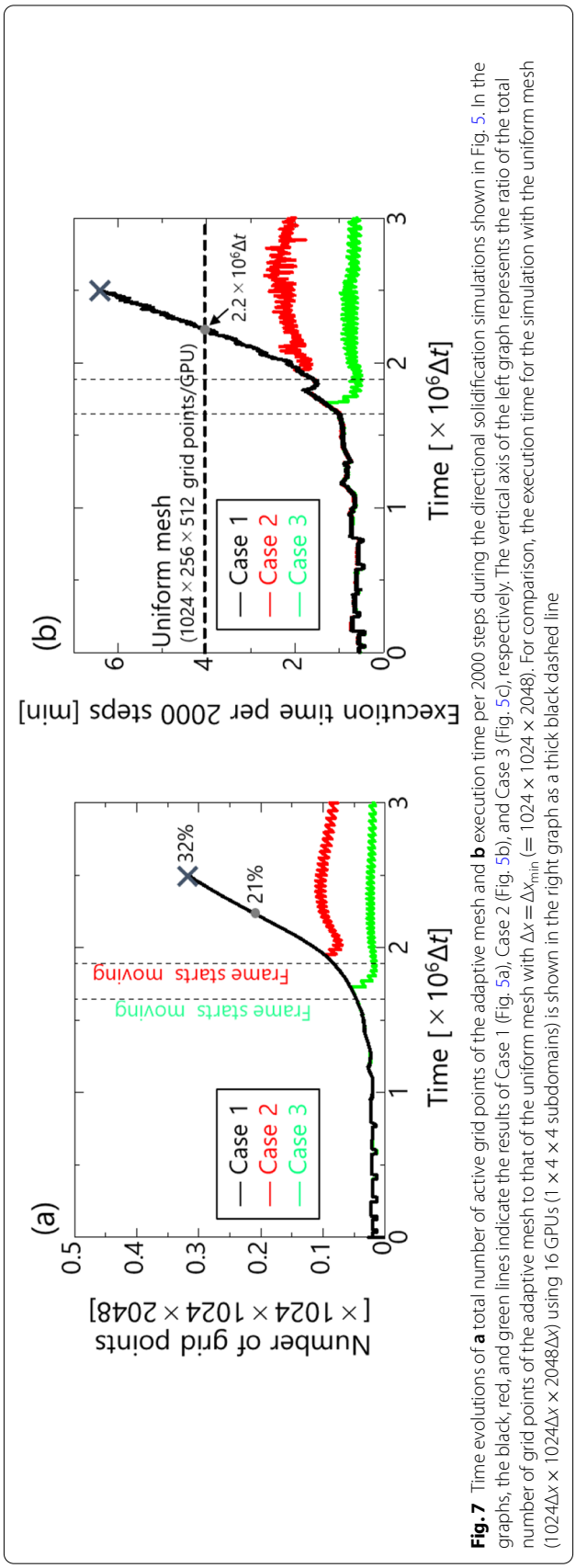

Fig. 7 Time evolutions of **a** total number of active grid points of the adaptive mesh and **b** execution time per 2000 steps during the directional solidification simulations shown in Fig. 5. In the graphs, the black, red, and green lines indicate the results of Case 1 (Fig. 5a), Case 2 (Fig. 5b), and Case 3 (Fig. 5c), respectively. The vertical axis of the left graph represents the ratio of the total number of grid points of the adaptive mesh to that of the uniform mesh with $\Delta x = \Delta x_{\min}$ (= 1024 × 1024 × 2048). For comparison, the execution time for the simulation with the uniform mesh ($1024\Delta x \times 1024\Delta x \times 2048\Delta x$) using 16 GPUs (1 × 4 × 4 subdomains) is shown in the right graph as a thick black dashed line

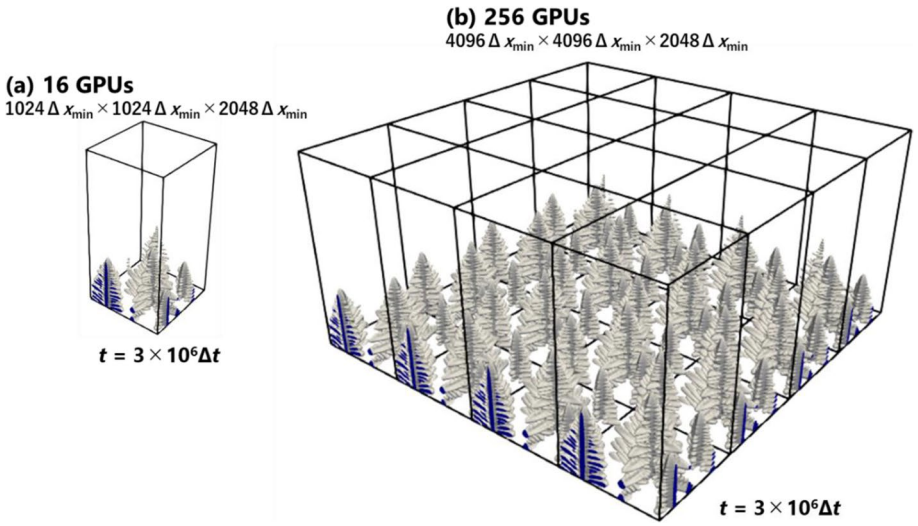

Fig. 8 Dendrite morphologies at $t = 3.0 \times 10^6 \Delta t$ used for weak scaling tests using **a** 16 and **b** 256 GPUs. In the weak scaling tests, the execution times for 2000 time steps from $t = 3.0 \times 10^6 \Delta t$ are measured. The distributions of φ and u at $t = 3.0 \times 10^6 \Delta t$ with 16 GPUs were obtained from the simulation shown in Fig. 5b. The distributions at $t = 3.0 \times 10^6 \Delta t$ with 256 GPUs were obtained by periodically arranging the results at $t = 3.0 \times 10^6 \Delta t$ in Fig. 5b

Figure 9 shows the parallel efficiency, which is expressed as the execution time per 2000 time steps from $t = 3.0 \times 10^6 \Delta t$. As shown in the figure, the execution time with 256 GPUs is only 1.7% longer than that with 16 GPUs. Thus, the execution time is almost constant for 16–256 GPUs, and good parallel computation performance is obtained. When 256 GPUs are used, a large-scale simulation of the growth of a massive number of dendrites with dimensions of $4096\Delta x_{\mathrm{min}} \times 4096\Delta x_{\mathrm{min}} \times 2048\Delta x_{\mathrm{min}}$ (Fig. 8b) can be performed with almost the same computational efficiency as that of the simulation with 16

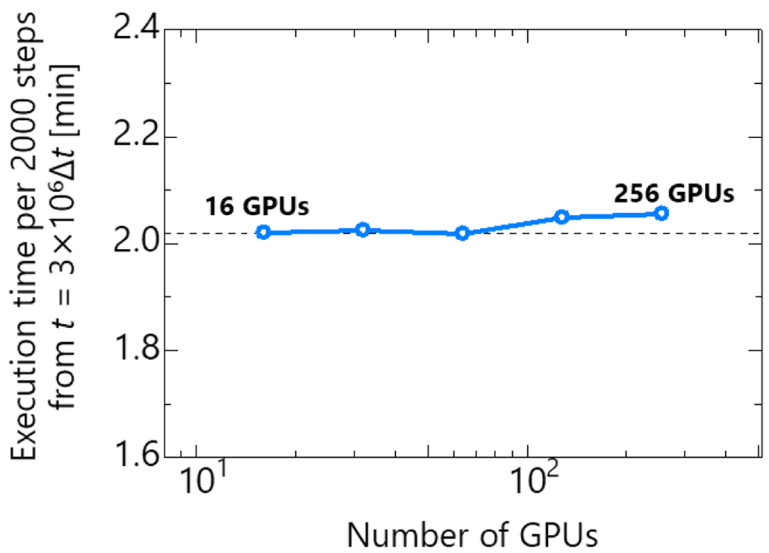

Fig. 9 Weak scaling test results of AMR-PF computation with multiple GPUs using the distributions shown in Fig. 8

GPUs. We conclude that the developed method is a powerful scheme for simulating the growth of numerous dendrites.

Conclusions

We developed a simulation scheme to improve the computational efficiency and scale of PF simulations for binary alloy solidification. The scheme employed multiple-GPU parallel computation and AMR. In the parallel implementation of multiple GPUs, we utilized domain subdivision considering dynamic load balancing. We evaluated the performance of the developed scheme by simulating columnar dendrite growth during the directional solidification of a binary alloy. The refinement conditions for AMR were evaluated through single-GPU computation, and it was confirmed that the computational accuracy was the same as that obtained using a uniform mesh. Next, we confirmed reasonable performance of dynamic load balancing in the domain subdivision through the multiple-GPU parallel simulations of directional solidification with the MFA. Finally, we performed weak scaling tests to evaluate the parallel computation performance and confirmed that good parallel computation performance was obtained for 16–256 GPUs.

The developed scheme can significantly improve the efficiency of simulations under solidification conditions with a large solute diffusion length and/or large primary arm spacing. Furthermore, in the future, we will apply this scheme to solidification conditions with melt convection and solid motion.

Abbreviations

PF : Phase-field; 2D: Two-dimensional; 3D : Three-dimensional; GPU : Graphics processing unit; AMR : Adaptive mesh refinement; CPU: Central processing unit; MFA: Moving frame algorithm.

Acknowledgements

Not applicable.

Authors' contributions

S.S. and T.T. designed and planned this study. S.S. implemented the simulation method supported by T.T. and T.A. T.T. and T.A. prepared the computational environment on TSUBAME3.0. S.S. performed the simulations, data processing, and visualization. S.S. interpreted the simulation results and wrote the manuscript with the help of T.T. and T.A. All authors read and approved the final manuscript.

Funding

This work was partly supported by KAKENHI Grant-in-Aid for Young Scientists (21K14041), Grant-in-Aid for Scientific Research (A) (20H00217), and Grant-in-Aid for Scientific Research (S) (19H05613) by the Japan Society for the Promotion of Science. This work was also supported by "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures" in Japan (Project ID: jh210010).

Availability of data and materials

The raw/processed data required to reproduce these findings cannot be shared at this time because the data are a part of an ongoing study.

Declarations

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Faculty of Mechanical Engineering, Kyoto Institute of Technology, Matsugasaki, Sakyo-ku, Kyoto 606-8585, Japan. [2]Global Scientific Information and Computing Center, Tokyo Institute of Technology, 2-12-1-i7-3, Ookayama, Meguro–ku, Tokyo 152-8550, Japan.

Received: 26 June 2021 Accepted: 1 December 2021
Published online: 06 January 2022

References

N. Al-Rawahi, G. Tryggvason, Numerical simulation of dendritic solidification with convection: Two-dimensional geometry. J. Comput. Phys. **180**, 471–496 (2002)

N. Al-Rawahi, G. Tryggvason, Numerical simulation of dendritic solidification with convection: Three-dimensional flow. J. Comput. Phys. **194**, 677–696 (2004)

B. Bellón, A.K. Boukellal, T. Isensee, O.M. Wellborn, K.P. Trumble, M.J.M. Krane, M.S. Titus, D. Tourret, J. Llorca, Multiscale prediction of microstructure length scales in metallic alloy casting. Acta Mater. **207**, 116686 (2021)

W.J. Boettinger, J.A. Warren, Simulation of the cell to plane front transition during directional solidification at high velocity. J. Cryst. Growth **200**, 583–591 (1999)

R.J. Borg, D.Y.F. Lai, Diffusion in α-Fe–Si alloys. J. Appl. Phys. **41**, 5193–5200 (1970)

S.G.R. Brown, T. Williams, J.A. Spittle, A cellular automaton model of the steady-state "free" growth of a non-isothermal dendrite. Acta Metall. Mater. **42**, 2893–2898 (1994)

F.P. Calderon, N. Sano, Y. Matsushita, Diffusion of managenese and silicon in liquid iron over the whole range of composition. Metall. Mater. Trans. B Process Metall. Mater. Process. Sci. **2**, 3325–3332 (1971)

R. Chen, Q. Xu, B. Liu, Cellular automaton simulation of three-dimensional dendrite growth in Al–7Si–Mg ternary aluminum alloys. Comput. Mater. Sci. **105**, 90–100 (2015)

A.J. Clarke, D. Tourret, Y. Song, S.D. Imhoff, P.J. Gibbs, J.W. Gibbs, K. Fezzaa, A. Karma, Microstructure selection in thin-sample directional solidification of an Al-Cu alloy: In situ X-ray imaging and phase-field simulations. Acta Mater. **129**, 203–216 (2017)

S. Cui, I.-H. Jung, Critical reassessment of the Fe-Si system. Calphad **56**, 108–125 (2017)

J.A. Dantzig, M. Rappaz, Solidification, (EPFL press, Lausanne, Switzerland, 2009)

H.J. Diepers, D. Ma, I. Steinbach, History effects during the selection of primary dendrite spacing. Comparison of phase-field simulations with experimental observations. J. Cryst. Growth **237-239**, 149–153 (2002)

K. Glasner, Nonlinear preconditioning for diffuse interfaces. J. Comput. Phys. **174**, 695–711 (2001)

T.Z. Gong, Y. Chen, Y.F. Cao, X.H. Kang, D.Z. Li, Fast simulations of a large number of crystals growth in centimeter-scale during alloy solidification via nonlinearly preconditioned quantitative phase-field formula. Comput. Mater. Sci. **147**, 338–352 (2018)

L. Gránásy, L. Rátkai, A. Szállás, B. Korbuly, G.I. Tóth, L. Környei, T. Pusztai, Phase-field modeling of polycrystalline solidification: From needle crystals to spherulites—A review. Metall. Mater. Trans. A **45**, 1694–1719 (2013)

M. Greenwood, K.N. Shampur, N. Ofori-Opoku, T. Pinomaa, L. Wang, S. Gurevich, N. Provatas, Quantitative 3D phase field modelling of solidification using next-generation adaptive mesh refinement. Comput. Mater. Sci. **142**, 153–171 (2018)

Z. Guo, J. Mi, S. Xiong, P.S. Grant, Phase field study of the tip operating state of a freely growing dendrite against convection using a novel parallel multigrid approach. J. Comput. Phys. **257**, 278–297 (2014)

Z. Guo, S.M. Xiong, On solving the 3-D phase field equations by employing a parallel-adaptive mesh refinement (Para-AMR) algorithm. Comput. Phys. Commun. **190**, 89–97 (2015)

D. Juric, G. Tryggvason, A front-tracking method for dendritic solidification. J. Comput. Phys. **123**, 127–148 (1996)

A. Karma, Y.H. Lee, M. Plapp, Three-dimensional dendrite-tip morphology at low undercooling. Phys. Rev. E **61**, 3996–4006 (2000)

A. Karma, W.-J. Rappel, Quantitative phase-field modeling of dendritic growth in two and three dimensions. Phys. Rev. E **57**, 4323–4349 (1998)

C. Karrasch, Solidification Kinetics in Undercooled Pure Iron and Iron-Boron Alloys under Different Fluid Flow Conditions (Doctoral Thesis) (Fakultät für Physik und Astronomie, Ruhr-Universität Bochum, 2016), p. 71

S.G. Kim, W.T. Kim, T. Suzuki, Phase-field model for binary alloys. Phys. Rev. E **60**, 7186–7197 (1999)

R. Kobayashi, A numerical approach to three-dimensional dendritic solidification. Exp. Math. **3**, 59–81 (1994)

W. Kurz, D.J. Fisher, R. Trivedi, Progress in modelling solidification microstructures in metals and alloys: Dendrites and cells from 1700 to 2000. Int. Mater. Rev. **64**, 311–354 (2018)

W. Kurz, M. Rappaz, R. Trivedi, Progress in modelling solidification microstructures in metals and alloys. Part II: Dendrites from 2001 to 2018. Int. Mater. Rev. **66**, 30–76 (2021)

C.W. Lan, Y.C. Chang, C.J. Shih, Adaptive phase field simulation of non-isothermal free dendritic growth of a binary alloy. Acta Mater. **51**, 1857–1869 (2003)

M. Ohno, Quantitative phase-field modeling of nonisothermal solidification in dilute multicomponent alloys with arbitrary diffusivities. Phys. Rev. E **86**, 051603 (2012)

M. Ohno, Quantitative phase-field modeling and simulations of solidification microstructures. ISIJ Int. **60**, 2745–2754 (2020)

M. Ohno, K. Matsuura, Quantitative phase-field modeling for dilute alloy solidification involving diffusion in the solid. Phys. Rev. E **79**, 031603 (2009)

M. Plapp, Phase-field modelling of solidification microstructures. J. Indian Inst. Sci. **96**, 179–198 (2016)

N. Provatas, N. Goldenfeld, J. Dantzig, Adaptive mesh refinement computation of solidification microstructures using dynamic data structures. J. Comput. Phys. **148**, 265–290 (1999)

S. Sakane, T. Takaki, M. Ohno, T. Shimokawabe, T. Aoki, GPU-accelerated 3D phase-field simulations of dendrite competitive growth during directional solidification of binary alloy. IOP Conf. Ser. Mater. Sci. Eng. **84**, 012063 (2015)

H.-Y. Schive, J.A. ZuHone, N.J. Goldbaum, M.J. Turk, M. Gaspari, C.-Y. Cheng, Gamer-2: A GPU-accelerated adaptive mesh refinement code – Accuracy, performance, and scalability. Mon. Not. R. Astron. Soc. **481**, 4815–4840 (2018)

T. Shimokawabe, T. Aoki, T. Takaki, T. Endo, A. Yamanaka, N. Maruyama, A. Nukada, S. Matsuoka, Peta-scale phase-field simulation for dendritic solidification on the TSUBAME 2.0 supercomputer, in Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, (ACM, New York, U.S., 2011), pp. 1–11

Y. Song, D. Tourret, F.L. Mota, J. Pereda, B. Billia, N. Bergeon, R. Trivedi, A. Karma, Thermal-field effects on interface dynamics and microstructure selection during alloy directional solidification. Acta Mater. **150**, 139–152 (2018)

T. Takaki, Phase-field modeling and simulations of dendrite growth. ISIJ Int. **54**, 437–444 (2014)

T. Takaki, T. Fukuoka, Y. Tomita, Phase-field simulation during directional solidification of a binary alloy using adaptive finite element method. J. Cryst. Growth **283**, 263–278 (2005)
T. Takaki, S. Sakane, M. Ohno, Y. Shibuta, T. Aoki, C.-A. Gandin, Competitive grain growth during directional solidification of a polycrystalline binary alloy: Three-dimensional large-scale phase-field study. Materialia **1**, 104–113 (2018)
T. Takaki, S. Sakane, M. Ohno, Y. Shibuta, T. Shimokawabe, T. Aoki, Large-scale phase-field studies of three-dimensional dendrite competitive growth at the converging grain boundary during directional solidification of a bicrystal binary alloy. ISIJ Int. **56**, 1427–1435 (2016a)
T. Takaki, S. Sakane, M. Ohno, Y. Shibuta, T. Shimokawabe, T. Aoki, Primary arm array during directional solidification of a single-crystal binary alloy: Large-scale phase-field study. Acta Mater. **118**, 230–243 (2016b)
D. Tourret, A. Karma, Growth competition of columnar dendritic grains: A phase-field study. Acta Mater. **82**, 64–83 (2015)
S. Tsuzuki, T. Aoki, Effective dynamic load balance using space-filling curves for large-scale SPH simulations on GPU-rich supercomputers, in Proceedings of 2016 7th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA), (IEEE, New York, U.S., 2016), pp. 1–8
S. Wang, Z. Guo, J. Kang, M. Zou, X. Li, A. Zhang, W. Du, W. Zhang, T.L. Lee, S. Xiong, J. Mi, 3D phase field modeling of multi-dendrites evolution in solidification and validation by synchrotron X-ray tomography. Materials (Basel) **14**, 520 (2021)
S. Watanabe, T. Aoki, Large-scale flow simulations using lattice Boltzmann method with AMR following free-surface on multiple GPUs. Comput. Phys. Commun. **264**, 107871 (2021)
S. Watanabe, T. Aoki, T. Takaki, A domain partitioning method using a multi-phase-field model for block-based AMR applications. Parallel Comput. **97**, 102647 (2020)
L. Wei, X. Lin, M. Wang, W. Huang, Orientation selection of equiaxed dendritic growth by three-dimensional cellular automaton model. Phys. B Condens. Matter **407**, 2471–2475 (2012)
A. Yamanaka, T. Aoki, S. Ogawa, T. Takaki, GPU-accelerated phase-field simulation of dendritic solidification in a binary alloy. J. Cryst. Growth **318**, 40–45 (2011)
A. Zhang, J. Du, J. Yang, Z. Guo, Q. Wang, B. Jiang, F. Pan, S. Xiong, General hierarchical structure to solve transport phenomena with dissimilar time scales: Application in large-scale three-dimensional thermosolutal phase-field problems. Phys. Rev. E **102**, 043313 (2020)
A. Zhang, Z. Guo, B. Jiang, J. Du, C. Wang, G. Huang, D. Zhang, F. Liu, S. Xiong, F. Pan, Multiphase and multiphysics modeling of dendrite growth and gas porosity evolution during solidification. Acta Mater. **214**, 117005 (2021b)
A. Zhang, B. Jiang, Z. Guo, J. Du, Q. Wang, F. Pan, S. Xiong, Solution to multiscale and multiphysics problems: A phase-field study of fully coupled thermal-solute-convection dendrite growth. Adv. Theory Simul. **4**(3), 2000251 (2021a)

Publisher's Note