Parallel computing for phase-field models

Alexander Vondrous, Michael Selzer, Johannes Hötzer and Britta Nestler

International Journal of High Performance Computing Applications published online 19 June 2013
DOI: 10.1177/1094342013490972

Article

International Journal of HIGH PERFORMANCE COMPUTING APPLICATIONS

The International Journal of High Performance Computing Applications
1–12
© The Author(s) 2013
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1094342013490972
hpc.sagepub.com
SAGE

Parallel computing for phase-field models

Alexander Vondrous[1], Michael Selzer[2], Johannes Hötzer[2] and Britta Nestler[2]

Abstract
The phase-field method is gradually exploiting new research areas, and the demand for the ability to simulate larger domains is increasing continuously with the move towards massive parallel computation. We emphasize the efficient usage of high-performance computing resources by investigating the scaling behavior of 1D domain decomposition, 3D domain decomposition and the runtime behavior of the commonly used moving simulation domain as well as 1D load balancing for a finite difference phase-field implementation. A simple performance model for blocking communication and measurements shows that it is necessary to apply 3D domain decomposition for a 3D domain to scale on high-performance clusters.

Keywords
Parallel computing, phase-field, performance analysis, microstructure simulation, finite differences domain decomposition

1 Introduction

Over the last two decades, the phase-field method has evolved into a powerful and important method to describe a broad variety of phenomena. It is most often applied to investigate solidification, solid-state transformation, coarsening or grain growth processes. Furthermore, this method delivers important insights in different scientific fields, for instance structure formation. It is, however, not limited to those fields, coupled effects with fluid flow, mechanical stress or magnetic fields have also been investigated. An overview is provided by Chen (2002). The development of the phase-field method continues but it is necessary to push the technical progress of the implemented models to exploit further research areas and to produce contributions, which otherwise would not be possible to calculate. However, the technical development is not only driven by the desire for knowledge, but also by the obligation to consciously and efficiently use existing and future processing resources. Computer scientists and hardware vendors are preparing to breach the exascale barrier by the end of this decade and prognosticate that processor development will continue in the direction of massive parallel data processing (Dongarra et al., 2009). Therefore, the phase-field community needs scalable algorithms and data structures that can keep pace with this general tendency.

The first phase-field model dates back to Langer, who investigated solidification of dendrites (Langer, 1986). Shortly after, it was proven that the phase-field method has favorable properties for tackling free boundary problems like the Stefan problem. One characteristic of the phase-field method is the use of a diffuse interface to describe the interface between material bodies with an order parameter (phase) without having to track the interface explicitly.

Investigations with more than two phases require more computation effort, because the derived evolution equations have to be computed for each phase separately. With the development of multiphase and multicomponent models (Steinbach et al., 1996; Nestler et al., 2005), the memory and computation barrier has been reached quickly using many phases. Gruber et al. (2006), Kim et al. (2006) and Vedantam and Patnaik (2006) contributed to overcoming this obstacle by recognizing that rarely more than six phases in three dimensions (3D) and five phases in two dimensions (2D) occur locally and need to be considered at a certain position in large grain-coarsening simulations. They introduced a dynamic phase vector to reduce memory consumption and calculation time significantly. The local reduction of the order parameter allows up to 2^{64} order parameters to be handled in the complete system with a

[1] Institute of Materials and Processes, Karlsruhe University of Applied Sciences, Germany
[2] Institute of Applied Materials - Reliability of Components and Systems, Karlsruhe Institute of Technology, Germany

Corresponding author:
Alexander Vondrous, Karlsruhe University of Applied Sciences Moltkestr. 30 Karlsruhe, 76133 Germany.
Email: alexander.vondrous@hs-karlsruhe.de

64-bit long integer to store the order parameter index. This concept of dynamically listing order parameters allows the investigation of large-scale 3D grain-coarsening processes covering valuable statical properties. Due to this modification, the applicability of the phase-field model is only restricted by the domain size. Therefore parallel computations using distributed memory are not only necessary to reduce computation time but are essential to provide the necessary amount of memory to handle large domain sizes.

However, if solidification is studied, diffusion processes play a central role e.g. as in binary or ternary eutectic phase transitions. The computations for the phase-field variables are coupled with heat and mass transfer equations, which is not necessary for pure grain-coarsening simulations. A lot of computational effort has to be applied for big simulations because a high grid resolution combined with a small timestep is mandatory. Adaptive mesh refinement (Provatas et al., 1999) and random walker (Plapp and Karma, 2000) algorithms are suitable for coping with the calculation-intense evolution equations.

George and Warren (2002) demonstrated, how to simulate a 3D dendrite with the phase-field method, one-dimensional (1D) domain decomposition and 1D load balancing. Current supercomputers allow large 3D simulations, which have been utilized by Shimokawabe et al. (2011) to simulate dendrite growth. They used a hybrid supercomputer with operating CPUs and general purpose graphics processing units (GPGPU) to perform a phase-field simulation with over 1 Petaflops (10^{15} floating point operations per second) for a grid of $4096 \times 6500 \times 10{,}400$ cells to simulate dendrite growth in 3D.

As you may see from this survey, many obstacles have already been overcome to simulate large domains and complex systems. We contribute to the technical development by pointing out the need of 3D domain decomposition for parallel computation of finite-differences-based 3D phase-field simulations on high-performance clusters employing the Message Passing Interface (MPI) standard (Message Passing Interface Forum, 2009). In addition, we present our performance model to estimate parallel efficiency, and show 1D load balancing and moving simulation domain measurements.

2 Phase-field implementation

PACE3D (Parallel Algorithms for Crystal Evolution in 3D) is a compilation of programs to simulate, analyze and visualize crystal growth and microstructure evolution. The phase-field model presented by Nestler et al. (2005) is the mathematical and physical basis for PACE3D and our investigations. In addition, phase transitions coupled with fluid dynamics, magnetic fields, elastic, plastic or volume-preserved models are implemented in PACE3D, but are not considered in the following investigation. The discussed implementation details give an overview of the relevant algorithms for the performance analysis.

The main workload of PACE3D is processed by the phase-field solver, the diffusion solver and its modules. The parallelized solver uses OpenMPI, an open-source implementation of the common MPI standard. Updates of the fields are calculated with the explicit Euler schema on a regular finite difference grid, such that we apply five-point stencil operations in 2D and seven-point stencil operations in 3D simulation domains. The cuboidal simulation domain is decomposed for parallel computation, by dividing it in 1D or in 3D as shown in Figure 1 into nearly equal sized subdomains, depending on the number of CPUs used.

Ghost cells contain the field values of the neighboring subdomain to enable the stencil operations and consist of a line of cells in 2D and a plane of cells in 3D. Each subdomain can be processed by one CPU for one timestep independently. After each timestep, the cells at the cutting plane are transferred to update the ghost cells.

In order to control and administrate the simulation, the Master–Worker pattern (Li and Malony, 2006) as illustrated in Figure 2 is implemented, so that each subdomain is assigned to one worker task. Each worker task is mapped to one CPU, and respectively to one CPU core. We do not distinguish between CPU cores and CPUs because each core is a completely independent CPU. The master task is responsible for initializing, controlling and stopping the worker tasks. Snapshots during a simulation are stored by the master in the 1D decomposition case, by receiving the subdomains of all worker tasks. For 3D domain decomposition, MPI I/O functions are used to write the snapshots of the simulation in parallel to reduce expensive I/O operations on the file system. Using a collective MPI I/O, all workers write the collected data in the same snapshot file.

The ghost cell update is performed with the help of blocking send and receive operations. It is a loose synchronization barrier, which involves every worker.

Fast workers with a smaller workload are waiting at the boundary update for slower workers with a bigger workload. This imbalance is caused by the implementation of phase-field update calculation and the infrastructure like hardware, operating system, etc. In this work, we only consider load imbalance caused by our implementation. Pure grain-coarsening simulations for example do not contain a bulk entropy density term in the entropy functional, such that only the phase-field updates have to be computed. The phase-field equations are solved only in interface regions between phases and not in the bulk area. Therefore no computation is performed if a cell and its neighbors have the same value. As a result, not every cell has to be computed. This leads to an imbalance of the workload in the simulation domain. In domain-decomposed parallel computed simulations, blocking communication between workers with an unbalanced workload produces unwanted CPU idle time.

To reduce the CPU idle time, a load-balancing mechanism is realized by dynamic domain decomposition (DDD) for the 1D domain decomposition case. Every worker measures its computation time for a defined number of timesteps and sends the accumulated computation time to the

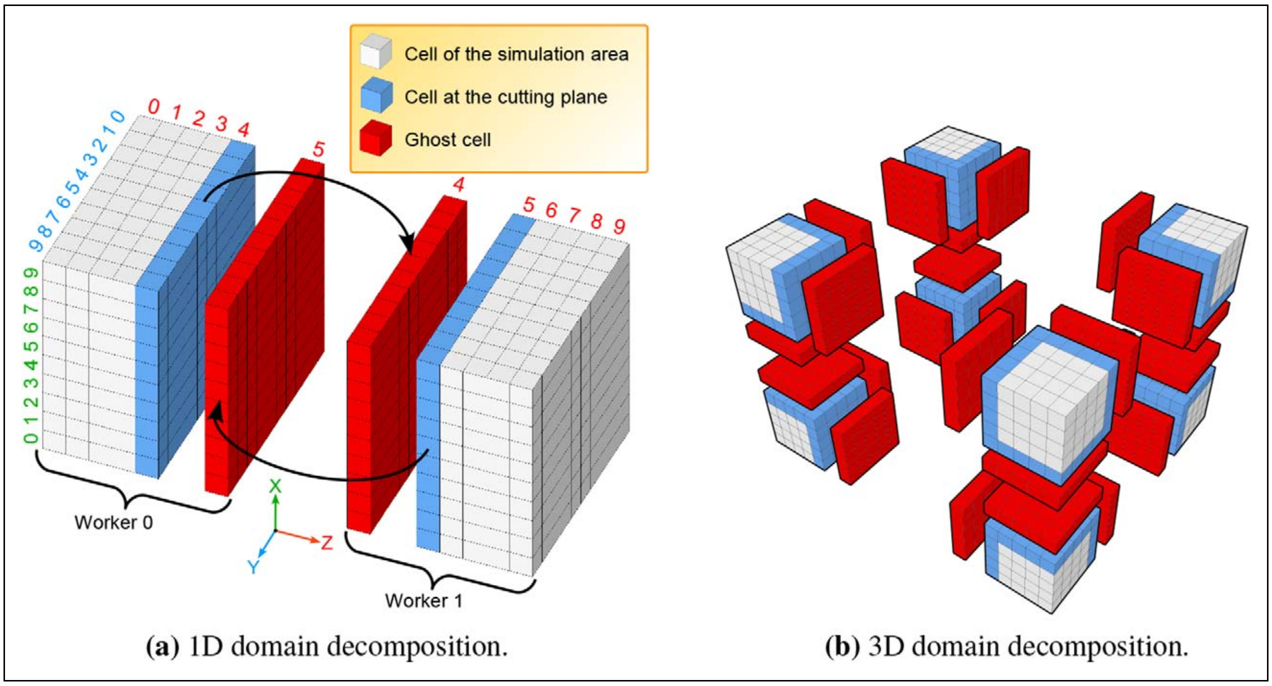

Cell of the simulation area
Cell at the cutting plane
Ghost cell
0 1 2 3 4
5
4
5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
0 1 2 3 4 5 6 7 8 9
X
Y
Z
Worker 0
Worker 1
(a) 1D domain decomposition.
(b) 3D domain decomposition.

Figure 1. 1D (a) and 3D (b) decomposition of a $10 \times 10 \times 10$ simulation domain. The domain is divided into two subdomains in 1D and eight subdomains in 3D. The red-colored ghost cells have to be updated by the neighboring CPU or core after each timestep to enable the seven-point stencil operation inside each subdomain cell.

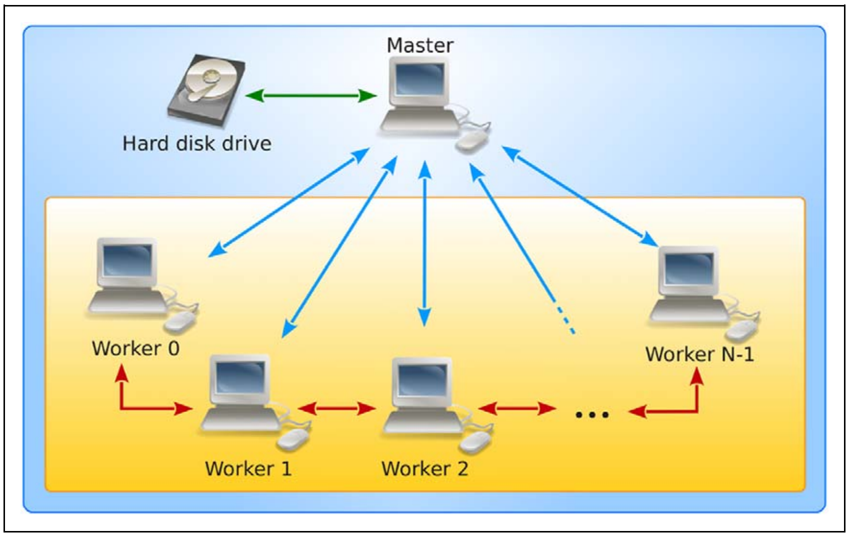


Figure 2. The master–worker pattern to organize and control the parallel simulation. The master controls all workers and stores the simulation results to the hard disk. Between timesteps, the workers update the ghost cells directly without an intermediate master .

master task. When all workers have sent their computation time, the master determines the optimal distribution of the domain and defines which worker has to enlarge or reduce its subdomain, such that all workers have a nearly equal workload. All workers obtain two integers, describing the number of layers they have to send or receive from their left and right neighbors. After receiving the number of layers, each worker has to determine how to enlarge, reduce or shift the subdomains layer-wise. Then all workers continue to compute the simulation loop with the new subdomain setup. The first load-balancing step is performed after the first timestep to obtain an approximation of the optimal domain distribution. The simulations in this study, which uses DDD, are configured to accumulate the computation time after every 100th timestep, which is an empirical value. It has been shown, that load balancing after each timestep can increase the runtime of a simulation, because some layers are always transferred back and forth between two workers and the communication between the master and the workers is an implicit barrier. In the opposite case,

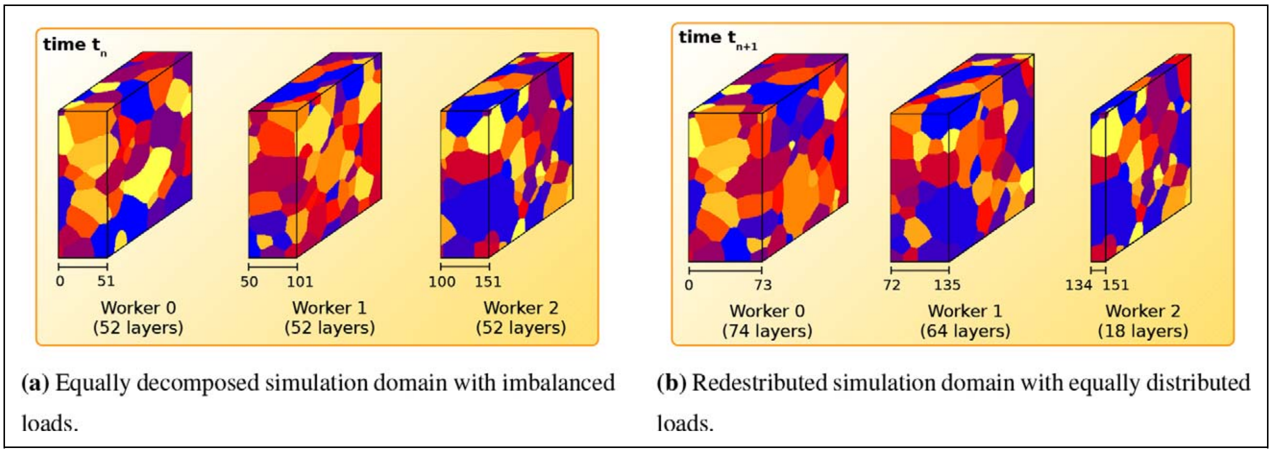


Figure 3. Subdomain size before and after load balancing. (a) At time t_n all workers have the same subdomain size (52 layers of *xy* planes). A high workload in the subdomain of Worker 2 leads to an imbalance. (b) The domain of Worker 2 is distributed to Workers 0 and 1. Worker 0 is the fastest worker and receives more layers than the slower Worker 1.

too few load-balancing steps lead to a suboptimal distribution of the load, thus the idle time can increase on some workers. Figure 3 illustrates the 1D load-balancing mechanism.

Another commonly used method to decrease computation time can only be applied if the area of interest moves along a trajectory during the simulation. Directional eutectic growth (Selzer et al., 2009) in a temperature gradient for example is a solidification process which moves from one side of the simulation domain to the other. A big simulation domain to capture the solidified area can be realized, however, it is not effective because a big part of the simulation domain has a small contribution but increases the simulation time and memory consumption. To avoid the increase of time and memory, a smaller simulation domain covering an area around the evolving phase boundary can be chosen, which moves with the solidification front. This tracking of the area of interest is referred to as a "Moving Box" algorithm. The Moving Box is only implemented for the 1D domain decomposition case, such that the movement direction is perpendicular to the decomposition axis to avoid communication through all workers for the domain movement. A plane is placed inside the domain to trigger the movement event. The trigger plane is also perpendicular to the decomposition axis and can be configured to react on a certain phase with a defined phase fraction. The additional communication effort after each timestep is introduced by a collective reduce operation to communicate a trigger event.

2.1 Performance modeling

To estimate the parallel runtime, speedup and efficiency, we create a general performance model incorporating 1D, 2D and 3D simulations. The simulation time is divided into calculation, communication and idle time, which are dependent on the subdomain size of the workers. Equation (1) describes the simulation time G^w for a worker w and over all timesteps $\tau_{\max}$ with the three terms related to computation, communication and idle time

$$G^w = \sum_{\tau=1}^{\tau_{\max}} \overbrace{n_c^\tau \cdot O^\tau}^{\text{Computation}} + \overbrace{\left(n_g^\tau + n_o^\tau\right) \cdot T^\tau}^{\text{Communication}} + \overbrace{I^\tau}^{\text{Idle}} \tag{1}$$

All parameters depend on the timestep τ in the general case. The computation time is estimated by the number of cells n_c^τ, in a worker's subdomain, multiplied by the average computation time per cell O^τ in the subdomain. To model the communication time, we assume domain decomposition, where after each timestep, all ghost cells n_g^τ of the neigboring workers are updated with the transfer time T^τ per cell. The communication time for load balancing is integrated by the number of cells n_o^τ, by which a domain is enlarged and reduced. Blocking communication together with load imbalance involves idle time I^τ, which we take into account via the last term.

This model allows the analysis of parallel scaling behavior incorporating the knowledge of time-dependent functions, which is not always given because of the dynamic simulation content. To estimate the effect of different domain decomposition methods, we simplify the model and assume that all parameters are independent of the timestep τ and that a cubic simulation domain of $n \times n \times n$ cells is used. Each worker has the same subdomain size, load balancing is not considered, and each computation cell in the simulation domain has the same computation time. Also, each communication cell is assumed to have the same communication time, such that idle time is able to be neglected.

We describe the sequential runtime G_{seq} on a single CPU as the total number of cells, multiplied by the calculation time per cell O and by the number of timesteps $\tau_{\max}$ according to the expression

$$G_{\text{seq}} = n_c \cdot O \cdot \tau_{\max} \tag{2}$$

For parallel execution (without load balancing and idle time), we add the communication term $n_g \cdot T \cdot \tau_{\max}$ reading

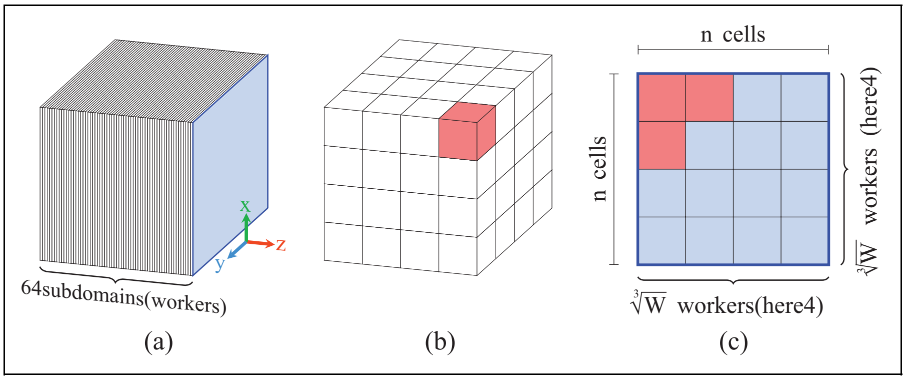


Figure 4. 1D and 3D decomposition with ghost layers for a simulation domain of ($n \times n \times n$) cells with 64 workers. (a) 1D decomposition along the z-axis with 64 workers to 64 subdomains. The blue area is one ghost surface. (b) 3D decomposition of the same domain with the same number of workers and three red ghost surfaces. (c) The ghost surfaces of the 1D and 3D decomposition are related to each other. Only $\frac{3}{16}$ of the 1D decomposition surface has to be updated in the case of 3D domain decomposition with 64 workers.

$$G^{w}_{\text{par}} = \left(n_{\text{c}} \cdot O + n_{\text{g}} \cdot T\right)\tau_{\max} \tag{3}$$

The ghost cell update between two neighbors is idealized to one send–receive operation, assuming an ideal full duplex connection between workers. In the case of 1D domain decomposition, each worker has to update ghost cells from its left and right subdomains, such that $n_{\text{g}} = 2 \cdot n^2$. The runtime for 2D domain decomposition is evaluated similarly to the 1D case. The four neighbors need to be updated with the fraction of the ghost cells of the 1D domain decomposition, resulting in the expression $n_{\text{g}} = 4 \cdot \frac{n^2}{\sqrt{W}}$. Following the 2D domain decomposition, the 3D case with six neighbors can be written as $n_{\text{g}} = 6 \cdot \frac{n^2}{\sqrt[3]{W^2}}$.

We constrain the domain size, so that it is equally divided by the number of workers W without reminder. Relying on these assumptions, each worker has the same runtime. To predict the scaling behavior, it is only necessary to model the runtime of one worker. Finally, the runtime models of a worker with 1D, 2D and 3D domain decomposition are

$$G_{1\text{D}} = \left(\frac{n^3}{W} \cdot O + 2 \cdot n^2 \cdot T\right)\tau_{\max} \tag{4}$$

$$G_{2\text{D}} = \left(\frac{n^3}{W} \cdot O + 4 \cdot \frac{n^2}{\sqrt{W}} \cdot T\right)\tau_{\max} \tag{5}$$

$$G_{3\text{D}} = \left(\frac{n^3}{W} \cdot O + 6 \cdot \frac{n^2}{\sqrt[3]{W^2}} \cdot T\right)\tau_{\max} \tag{6}$$

To avoid invalid configurations, we restrict the domain size and the number of workers, such that the results of $\frac{n^3}{W}$, $\sqrt[3]{W}$ and $\sqrt{W}$ are integers. This restriction limits the domain extensions and the number of workers, but it allows an easier investigation of parallel scaling behavior.

In order to estimate the scaling behavior of a domain with dimension Ω and a d-dimensional domain decomposition, we derive the generalized formulation of equation (3), proceeding equations (4)–(6), to describe the simulation runtime by

$$G^{\Omega}_{d} = \left(\frac{n^{\Omega}}{W} \cdot O + 2 \cdot d\left(\frac{n}{\sqrt[d]{W}}\right)^{d-1} \cdot n^{\Omega-d} \cdot T\right)\tau_{\max} \tag{7}$$

with the restriction $\Omega \geq d$.

Figure 4 compares the 1D and 3D decomposition methods with 64 workers operating on a cubic domain of $n \times n \times n$ cells to explain the origin of the restrictions and terms of the models. Figure 4(c) compares the number of cutting plane cells in a higher order direction to illustrate the communication advantage of the 3D decomposition method for 3D domains. Each worker has the same number of cells to compute, but a different number of boundary cells to transfer.

To compare the efficiency of the decomposition methods dependent on the worker count, we derive the general function $E(W)^{\Omega}_{d}$ based on equation (7) with the common definition of speedup $\left(S = \frac{G_{\text{seq}}}{G_{\text{par}}}\right)$ and efficiency $\left(E = \frac{S}{W}\right)$

$$\begin{aligned} E(W)^{\Omega}_{d} &= \frac{n^{\Omega} \cdot O}{\left(\frac{n^{\Omega}}{W} \cdot O + 2 \cdot d \cdot \left(\frac{n}{\sqrt[d]{W}}\right)^{d-1} \cdot n^{\Omega-d} \cdot T\right) W} \\ &= \frac{1}{1 + \frac{2d}{n} \cdot \frac{T}{O} \cdot \sqrt[d]{W}} \end{aligned} \tag{8}$$

3 Simulation settings for efficiency estimations

Different applications are better suited to certain optimization techniques. Therefore, we define different microstructure morphologies such as dendrite growth (Wesner et al., 2012), solidification of ternary eutectics (Choudhury et al., 2011) and grain coarsening in 3D (Nestler et al., 2008) to measure the performance of the implemented optimization techniques. The three examples cover representative classes of phase-field applications with long-range diffusion control, short-range diffusion control with low-

Table 1. Hardware and software specifications of the clusters utilized in this work.

Hardware	IMP Cluster	XC 4000
Number of compute nodes	48	750
Cores per node	8 Cores (2 AMD Quad-Core Opterons)	4 Cores (2 AMD Dual-Core Opterons)
Total cores	384	3000
Clock cycle	2 GHz	2.6 GHz
Main memory per core	2 GByte	4 GByte
Node interconnect	4X DDR Infiniband	4X DDR Infiniband
Interconnect bandwidth	16 GBit/s	16 GBit/s
Interconnect topology	Fat tree	Fat tree
Software		
Operating system	Debian Linux	Red Hat Linux
Compiler	GCC 4.3.2	Intel Compiler 10.1
MPI	OpenMPI 1.3.4	OpenMPI 1.5.1

order junctions and surface-driven coarsening with a strongly varying number of high-order junctions. The review article by Chen (2002) describes the major phase-field applications and references literature for each application for further reading.

Considering parallel execution scalability, speedup and efficiency is of interest. Speedup is calculated by dividing the sequential execution time by the parallel execution time. If the number of used CPUs increases, more time is spent on e.g. synchronization. These "frictional losses" can be seen in the efficiency of a parallel program. The efficiency is calculated by dividing the speedup by the number of CPUs.

If information on the sequential and parallel execution times is gathered to obtain a performance description, the system specification (operating system, hardware properties, etc.) is necessary in order to make a well-founded comparison. The first HPC used in this work belongs to the Institute of Materials and Processes of the University of Applied Sciences, Karlsruhe and the second belongs to the Scientific Computing Center of the Karlsruhe Institute of Technology. The hardware and software specifications used are listed in Table 1.

To evaluate the computational efficiency of the different optimization algorithms and strategies, we define a set of three typical microstructure simulations with different requirements and characteristics for computational treatment. The considered cases are labeled with A, B and C in order to refer precisely to the simulation setup in the subsequent sections.

3.1 Dendrite morphology

Thermal, as well as solutal dendrite growth is a long-range diffusion-controlled process. In addition to the surface energy density contributions in the free energy functional, a chemical potential formulation has to be introduced to describe the thermodynamical bulk terms of the different phases in a binary or higher component alloy system. The diffusion processes of the alloy components were computed using an additional evolution equation, as shown in Wesner et al. (2012). During the evolution of equiaxed dendritic structures, a complex surface geometry with a main trunk, tip properties, and secondary and ternary side branches is formed. We configure two dendrite simulation environments of a 2D and 3D domain.

1. A.1. 2D domain of 1000×1000 cells with one nucleus in the center
2. A.2. 3D domain of $200 \times 200 \times 200$ cells with one nucleus in the corner
3. A.3. 3D domain of $500 \times 500 \times 500$ cells with one nucleus in the corner

In A.1 we set up a 2D simulation of a single Ni-rich dendrite in a Ni–Cu melt with solutal solidification as in Warren and Boettinger (1995) of 1000×1000 cells to compare the runtime between the IMP cluster and the XC 4000 cluster.

The same Ni–Cu system is used in A.2 for a single 3D dendrite to compare the parallel scalability of 1D and 3D domain decomposition due to small variations in computation time per cell. The evolution equations for the concentration fields are solved in each cell and take more computation time in contrast to solving the phase-field evolution equations, which are only computed for the interface region. A.3 is only computed with 1D domain decomposition to compare 1D domain decomposition between A.3 and the 2D domain A.1.

We use a small domain of 200^3 cells to obtain the sequential runtime within a reasonable timeframe. Larger simulation domains can be simulated, however, the runtime increases up to months for the sequential computation. To further reduce the computational effort, we make use of the dendrite symmetry and place the nucleus in one corner of the domain. Figure 5(a) shows the resulting 3D dendrite after mirroring along the symmetry planes.

3.2 Directional solidification of a ternary eutectic system

Eutectic growth is a diffusion-controlled process with a short-range diffusion zone involving the formation of heterogeneous microstructures with different solid phases and

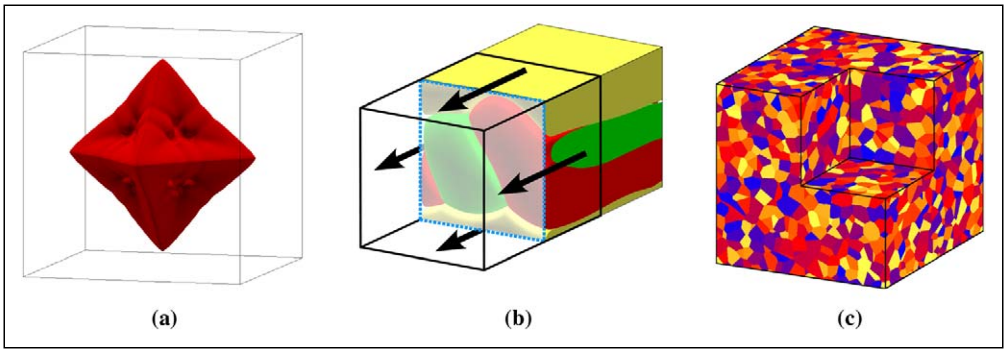


Figure 5. Illustration of the three simulation cases: (a) reconstructed 3D dendrite of configuration A.2; (b) Moving Box of configuration B.4; and (c) Voronoi tesselation generated according to the starting condition for C.1.

moving triple and quadruple junctions. A precise investigation of pattern characteristics is provided by Choudhury et al. (2011) and a theoretical discussion of the junctions can be found in Folch and Plapp (2005). In directional solidification, most of the effects such as diffusion and phase transformation take place in regions around the solid–liquid front. Therefore, a high grid resolution is needed at the solidification front combined with an increased numerical effort. A ternary eutectic solidification of three distinct solid phases (model system) grows in a dedicated direction as depicted in Figure 5(b). We configure four ternary eutectic growth simulations.

1. B.1. 3D domain of $300\times100\times100$ cells
2. B.2. 3D domain of $100\times100\times300$ cells
3. B.3. 3D domain of $100\times100\times300$ cells with load balancing
4. B.4. 3D domain of $100\times100\times100$ cells with Moving Box

Each configuration is executed with 100 workers to obtain comparable runtime performances. The differences between the four configurations include the domain orientation, domain size and the activated optimizations, thus the resulting eutectics are always equal except for the Moving Box configuration. Physical dynamics outside of the Moving Box are not captured, such that the results are not exactly equal.

By decomposing the domain along the z-axis for parallel computation, the domain orientation changes the number of ghost cells. We determine the influence of the communication time on the total runtime. In the worst case (B.1) 300×100 cells and in the best cases (B.2–B.4) 100×100 cells have to be updated after each timestep, respectively.

3.3 Grain growth in a polycrystalline system

For applications related to grain structure evolution, the free energy functional consists of the gradient and potential energy density terms (Nestler et al., 2008). Grain coarsening is a process inherently driven by curvature minimization and is therefore less computationally intensive compared to diffusion-controlled processes. As coarsening proceeds, the grain boundaries reduce, leading to an uneven load in the computing cluster, and making it a suitable example for measuring the effect of dynamic domain decomposition.

1. C. 1. 3D domain of $500\times500\times500$ cells
2. C. 2. 3D domain of $250\times500\times1000$ cells

The starting condition is generated randomly using the Voronoi algorithm as depicted in Figure 5(c) for C.1. One hundred and twenty five workers calculate both configurations with and without load balancing. In contrast to the dendrite simulations, we set the maximum runtime to three days.

4 Results

In this section, we outline the measurement results for scalability, performance and dynamic load balancing for the sample settings A, B and C.

4.1 Scalability through domain decomposition

Scaling behavior in the area of high-performance computing is characterized as *weak*, when the problem size per CPU stays constant and the number of CPUs increases, or as *strong*, when the total problem size stays constant and the number of CPUs increases. We demonstrate strong scaling behavior for A.2 with 1D and 3D domain decomposition in the efficiency curves in Figure 6(b). The thin black lines denote the least-squares-fit of equation (8) with $\Omega = 3, d = 1$ and $d = 3$ for a 3D domain with 1D and 3D domain decomposition. The parameters for a 2D domain with 1D domain decomposition are $\Omega = 2$ and $d = 1$.

The communication-to-computation coefficient $\frac{T}{O}$ is the only unknown parameter, which we fit with the least-squares-fit method to extrapolate and estimate scaling behavior. Figure 6(b) shows the scaling behavior of A.1 and A.3 with 1D domain decomposition, and an estimation for 3D and 2D domain decomposition, respectively, based on estimated communication-to-computation coefficients. The sequential and parallel 2D dendrite simulations for A.1 were carried out on the IMP and the XC 4000 (short

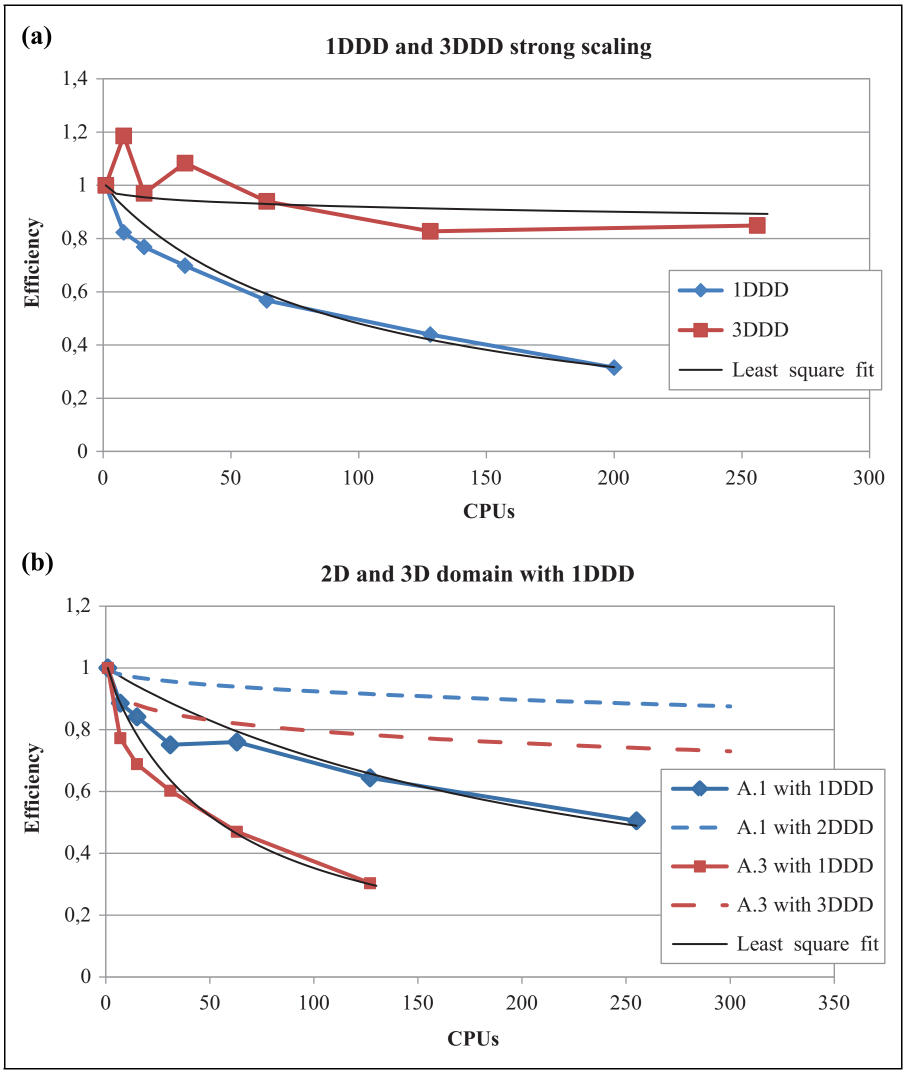


Figure 6. Measured strong scaling behavior of A.1, A.2 and A.3. (a) 1D and 3D domain decomposition of A.2. (b) 1D domain decomposition of A.1 and A.3. Dashed lines denote the estimated scaling behavior with 3D domain decomposition.

Table 2. Runtime of the 1000 $\times$ 1000 cells 2D dendrite simulation A.1 on the IMP and XC 4000 cluster with varying CPU count and 1D decomposition.

Workers (cores)	IMP hh:mm:ss	XC hh:mm:ss	IMP/XC
1 (1)	70:49:57	58:43:05	120.6%
8 (9)	11:25:16	09:30:17	120.2%
16 (17)	05:36:37	04:31:25	124.0%
32 (33)	03:02:31	02:16:06	134.1%
64 (65)	01:28:44	01:11:17	124.5%
128 (129)	00:51:56	00:39:05	132.9%
256 (257)	00:32:59	00:23:09	142.5%

XC) cluster. By computing the same simulations multiple times and taking the mean value, the measurement noise can be reduced. However, this step was not conducted due to resource protection. Table 2 lists the runtime data for both clusters. The simulations on the IMP cluster needed, on average, 28.4% longer.

These measurements reflect a small part of the scaling behavior. It is of interest to determine the overall scaling performance of the 1D and 3D decomposition methods for clusters with more than 100,000 CPUs. Based on the performance model presented, we plot the efficiency surface over the number of CPUs and over the communication-to-computation coefficient $\frac{T}{O}$ as prescribed by equation (7) for a 3D domain ($\Omega = 3$) with 1D ($d = 1$) and 3D ($d = 3$) domain decomposition. The aims of using large systems include the ability to simulate complex problems, to resolve phenomena with a higher resolution and to describe scale-bridging effects. We assume a cubic domain of 5000 $\times$ 5000 $\times$ 5000 cells to estimate the scaling behavior of 1D and 3D domain decomposition in large computational domains and plot the results in Figure 7.

These predictions are only valid if the communication-to-computation coefficients $\frac{T}{O}$ are large enough. If the network is very fast, then T is small and it makes almost no difference which decomposition method is used. The only difference would be the number of usable processors.

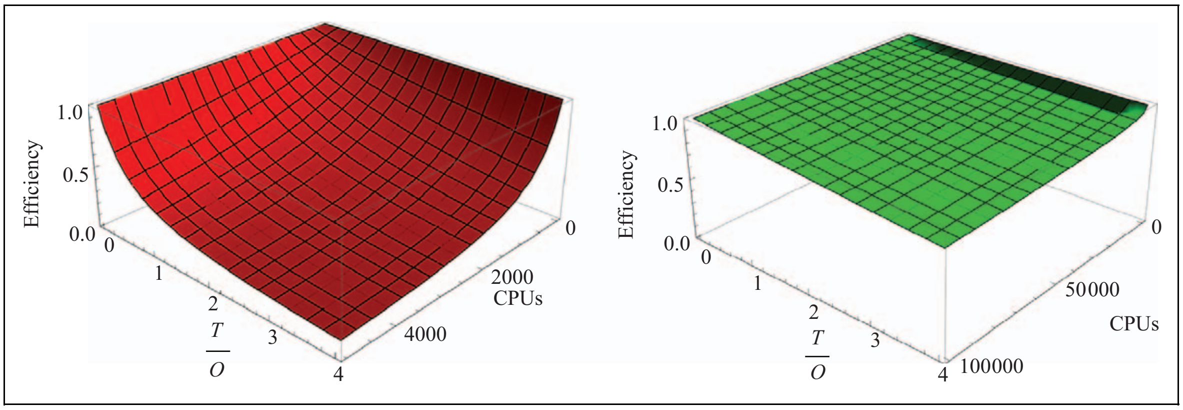


Figure 7. Efficiency of 1D and 3D domain decomposition methods with a domain of 5000 × 5000 × 5000 cells. (a) The efficiency surface of the 1D domain decomposition method is shown for up to 5000 CPUs, because 1D decomposition limits the number of usable CPUs. (b) The efficiency surface of the 3D domain decomposition method enables the employment of larger clusters. The image refers to 100,000 CPUs.

Table 3. Estimated and calculated communication-to-computation coefficients $\frac{T}{O}$ for 1D and 3D domain decomposition (short 1DDD and 3DDD).

Setup	1DDD	3DDD
A.1	2.05	—
A.2	1.08	0.63
A.3	4.59	—
B.1 & B.2	0.78	—

Table 3 contains the five calculated coefficients based on the measurements. The coefficients for B.1 and B.2 are calculated by solving two linear equations, based on equation (4) without the cubic domain restriction.

4.2 Communication bound

If a non-cubic domain is used, the performance depends on the choice of the domain orientation. The ternary eutectic simulation B.1 has a runtime of 240,760 seconds. The rotated domain B.2 has a runtime of 142,731 seconds, which is 59.28% of the non-rotated domain B.1. With dynamic domain decomposition after every 100th timestep (configuration B.3), the runtime decreases to 129,394 seconds, which is 53.74% of B.1. Applying the Moving Box algorithm to B.4, the runtime decreases to 84,809 seconds, which is 35.23% of B.1. These timings are summarized in Table 3 and indicate the communication-bound character of the simulations.

4.3 Load balancing

The runtime measurements of C.1 and C.2 with and without dynamic domain decomposition are shown in Figure 8. The time to calculate 100 timesteps is accumulated to one data point in the diagram. The elongated domain of C.2 seems to have a higher imbalance than the cubic domain.

5 Discussion

The measurements demonstrate the communication-bound character of the implementation, and show improvements to 3D domain decomposition, load balancing and the Moving Box algorithm.

Comparing the runtime of the 2D dendrite simulation A.1 between the IMP and XC cluster, we assume that the CPU frequency leads to a better performance on the XC 4000 cluster because the clock cycle is the most obvious difference between the systems. Considering the clock frequency, the CPUs of the IMP system need 30% more time than the CPUs of the XC cluster. Another issue is the configuration of the Infiniband network connecting the computing nodes and the file system, which could explain the performance gain of the XC cluster with many CPUs.

5.1 Scalability through domain decomposition

The strong scaling measurements in Figure 6(a) illustrate a faster efficiency drop for 1D than for 3D domain decomposition, which we relate to the communication-bound performance. Oscillations in the efficiency for 3D domain decomposition can be observed, which are not visible for higher numbers of CPUs. This can be explained by cache effects or a non-optimal sequential implementation as mentioned by Helmbold (1990). Another impact on the performance, leading to oscillation, is I/O of the snapshots. I/O is traffic-dependent and occurs randomly, possibly leading to a busy file system, to slow down the runtime considerably. Investigations to clarify the reasons for an efficiency greater than one have to be carried out to exploit the performance gain for all simulations.

The least-squares fit of the performance model deviates from the measurements if fewer than 50 CPUs are used for all 1D domain decomposition simulations in Figure 6. The reason for the deviation is the uneven distributed load between the workers for the simulations performed. The model assumption of an equally distributed load is not

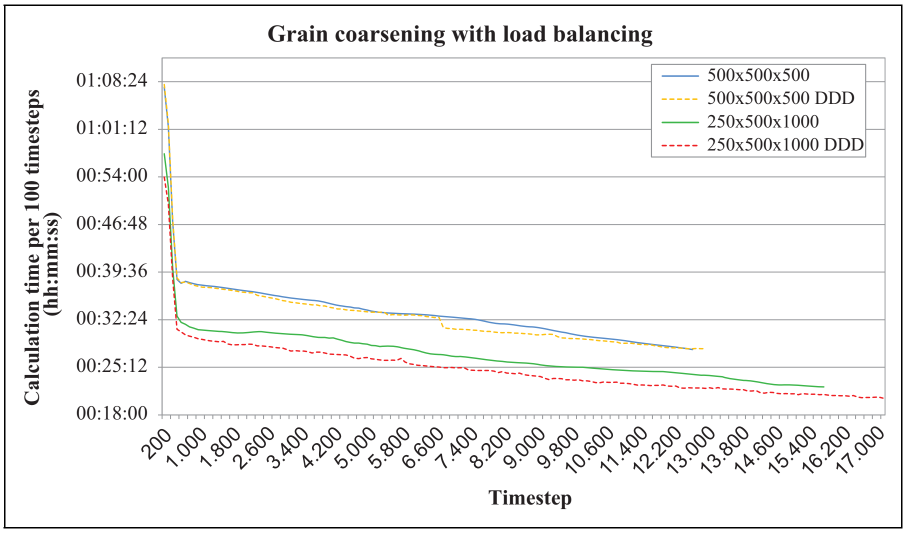


Figure 8. Grain-coarsening simulations C.1 and C.2 with a runtime limit of three days. The dashed lines denote the behavior with active load balancing.

valid. The load of the dendrite simulations is not distributed equally, such that a decomposition with fewer than 50 CPUs contains workers not contributing to computing the load. Hence, the efficiency drops below the model prediction. Fewer workers result in a bigger deviation from the model. Load-balancing mechanisms are the solution to avoid the efficiency drop and to provide an even load distribution.

The efficiency drop with 1D domain decomposition occurs faster for 3D domains than for 2D domains as shown in Figure 6(b). This is a result of the relation between the dimensionality of the decomposition and the dimensionality of the domain as proposed by equation (8). If the dimensionality of the decomposition is closer to the dimensionality of the domain, a higher efficiency can be obtained for the same simulation. Figure 7 shows a significantly different scaling behavior between 1D and 3D domain decomposition, as well as the limits of 1D domain decomposition for a 3D domain.

5.2 Communication bound

The influence of the domain orientation on the runtime is also significant and can be observed within the ternary eutectic simulations B.1–B.4. The simulation of B.2 required only 59% of the simulation time of B.1. The main cause of the runtime differences can be related to communication, because the computation effort for B.1 and B.2 is the same. The boundary layer size is 100×100 for B.2 and 300×100 for B.1, respectively. In the latter case, the number of cells, which have to be transferred within the network is three times larger for each computed timestep of the iteration, thus leading to at least three times longer communication times. Another reason for the speedup is a reduced number of cache misses due to better memory alignment. Dynamic domain decomposition in B.3 further improves the runtime. The optimization of only calculating the evolution equations for cells without the same neighbors seems to have an influence on the runtime, even with the long-range concentration field. Comparing the Moving Box simulation B.4 with B.1, the runtime is almost $\frac{1}{3}$, in accordance with expectations. One third fewer computation cells and $\frac{1}{3}$ less communication leads to $\frac{1}{3}$ of the runtime.

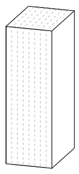
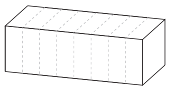
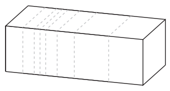
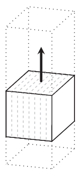
5.3 Load balancing

Load balancing shows that the benefit strongly depends on the simulation content. Figure 7 indicates only one significant load imbalance for C.1 in a cubic domain ($500 \times 500 \times 500$) at about timestep 6800. The load distributes after the imbalance homogeneously until about timestep 13,000, which explains the convergence. The elongated domain C.2 ($250 \times 500 \times 1000$) has the advantage of moving smaller pieces of work, which enables finer and hence better load balancing. All measurements contain a high computation time at the beginning, which is the result of interface development during the first 100 timesteps. Further investigations have to be carried out to identify the best configuration setups and to determine the limits of the dynamic domain decomposition method for 1D domain decomposition.

6 Conclusion

All measurements combined draw a rough performance picture of the implemented phase-field model with a clear direction for further improvement. The measurements indicate the communication-bound character of the implementation, such that a reduction of the communication time is one important step to achieving good strong scaling behavior on large high-performance systems. Extrapolations with the performance model and the strong scaling measurements show that 3D domain decomposition reduces communication time, hence increases the efficiency

Table 4. Runtime of the 3D ternary eutectic configurations B.1–B.4.

Configuration	 B.1	 B.2	 B.3	 B.4
Domain size	300 × 100 × 100	100 × 100 × 300	100 × 100 × 300	100 × 100 × 100
Optimization	—	—	Dynamic Domain Decomposition	Moving Box
Runtime (d:hh:mm:ss)	2:18:52:40	1:15:38:51	1:11:56:34	0:23:33:29
Runtime (percentage)	100%	59.28%	53.74%	35.23%

significantly. If the communication-to-computation coefficient $\frac{T}{O}$ is known, the scaling behavior can be predicted to a certain extent.

Besides the implementation details, it is of importance which hardware is utilized for the simulation tasks. We investigated the runtime on two similar clusters and found different runtime behavior, which is mainly caused by the clock frequency of the CPUs.

One-dimensional load balancing reduces simulation time but the benefit is strongly dependent on the simulation content and the degree of freedom. The load imbalance or the degree of freedom do not seem to be very high for the simulations performed. A major increase in load imbalance can be realized with the help of an adaptive mesh refinement scheme and with 3D domain decomposition. A proper investigation of load-balancing strategies has to be performed to enable general statements about the reachable benefit to be made.

To obtain deeper insight into the simulation runtime on high-performance computers and to fine tune the implementation, it will be necessary to apply elaborated profiling tools like VampirTrace (Müller et al., 2007) or Scalasca (Geimer et al., 2010). The next steps to improve the scalability, in addition to 3D domain decomposition, are to implement a communication-hiding mechanism, 3D load balancing and adaptive mesh refinement.

Acknowledgements

The phase-field simulations presented in this work were performed on the Landeshöchstleistungsrechner XC 4000 of the Karlsruhe Institute of Technology and on the ICE cluster funded by the Baden–Württemberg Stiftung. The authors gratefully acknowledge access to these systems.

Funding

We thank the DFG for funding our investigations in the framework of the Graduate School 1483 and the Ministry of the State Baden–Württemberg for support within "kooperatives Promotionskolleg" in Karlsruhe.

References

Chen LQ (2002) Phase-field models for microstructure evolution. *Annual Review of Materials Research* 32(1): 113–140.

Choudhury A, Plapp M, and Nestler B (2011) Theoretical and numerical study of lamellar eutectic three-phase growth in ternary alloys. *Physical Review E* 83(5): 051608.

Dongarra J, Beckman P, Aerts P, Cappello F, Lippert T, Matsuoka S, et al. (2009) The International Exascale Software Project: a call to cooperative action by the global high-performance community. *International Journal of High Performance Computing Applications* 23(4): 309–322.

Folch R and Plapp M (2005) Quantitative phase-field modeling of two-phase growth. *Physical Review E* 72(1): 011602.

Geimer M, Wolf F, Wylie BJN, Ábrahám E, Becker D, and Mohr B (2010) The Scalasca performance toolset architecture. *Concurrency and Computation: Practice and Experience* 22(6): 702–719.

George W and Warren J (2002) A parallel 3D dendritic growth simulator using the phase-field method. *Journal of Computational Physics* 177(2): 264–283.

Gruber J, Ma N, Wang Y, Rollett AD, and Rohrer GS (2006) Sparse data structure and algorithm for the phase field method. *Modelling and Simulation in Materials Science and Engineering Systems* 14(7): 1189–1195.

Helmbold D (1990) Modelling speedup (n) greater than n. *IEEE Transactions on Parallel and Distributed Systems* 1(2): 250–256.

Kim S, Kim D, Kim W, and Park Y (2006) Computer simulations of two-dimensional and three-dimensional ideal grain growth. *Physical Review E* 74(6): 061605.

Langer JS (1986) Models of pattern formation in first order phase transitions in directions in condensed matter. *World Scientific Singapore* 165(1): 165–186.

Li L and Malony A (2006) Model-based performance diagnosis of master–worker parallel computations. In: Nagel W, Walter W, and Lehner W (eds) *Lecture Notes in Computer Science*. Berlin: Springer, vol. 4128, pp. 35–46.

Message Passing Interface Forum (2009) *MPI: A Message-Passing Interface Standard, Version 2.2*. Stuttgart: High-Performance Computing Center.

Müller MS, Knüpfer A, Jurenz M, Lieber M, Brunst H, Mix H, et al. (2007) Developing scalable applications with Vampir, VampirServer and VampirTrace. In: Bischof C, Bücker M, Gibbon P, Joubert GR, Lippert T, and Mohr B et al. (eds) *Advances in Parallel Computing*. Amsterdam: IOS Press, vol. 15, pp. 637–644.

Nestler B, Garcke H, and Stinner B (2005) Multicomponent alloy solidification: Phase-field modeling and simulations. *Physical Review E* 71(4): 041609.

Nestler B, Reichardt M, and Selzer M (2008) *Massive Multi-Phase-Field Simulations*. Weinheim, Germany: Wiley-VCH, pp. 1251–1255.

Plapp M and Karma A (2000) Multiscale random-walk algorithm for simulating interfacial pattern formation. *Physical Review Letters* 84(8): 1740–1743.

Provatas N, Goldenfeld N, and Dantzig J (1999) Adaptive mesh refinement computation of solidification microstructures using dynamic data structures. *Journal of Computational Physics* 148(1): 265–290.

Selzer M, Jainta M, and Nestler B (2009) A Lattice–Boltzmann model to simulate the growth of dendritic and eutectic microstructures under the influence of fluid flow. *Physica Status Solidi B* 246(6): 1197–1205.

Shimokawabe T, Aoki T, Takaki T, Endo T, Yamanaka A,, Maruyama N, et al. (2011) Peta-scale phase-field simulation for dendritic solidification on the TSUBAME 2.0 supercomputer. In: *2011 international conference for high performance computing, networking, storage and analysis*, Seattle, WA, USA, 12–18 November 2011, pp. 1–11. New York: ACM Press.

Steinbach I, Pezzolla F, Nestler B, Seeßelberg M, Prieler R, Schmitz G, et al. (1996) A phase field concept for multiphase systems. *Physica D: Nonlinear Phenomena* 94(3): 135–147.

Vedantam S and Patnaik BSV (2006) Efficient numerical algorithm for multiphase field simulations. *Physical Review E* 73(1): 016703.

Warren J and Boettinger W (1995) Prediction of dendritic growth and microsegregation patterns in a binary alloy using the phase-field method. *Acta Metallurgica et Materialia* 43(2): 689–703.

Wesner E, Choudhury A, August A, Berghoff M, and Nestler B (2012) A phase-field study of large-scale dendrite fragmentation in Al–Cu. *Journal of Crystal Growth* 359: 107–121.

Author biographies

Alexander Vondrous is a PhD student at the Karlsruhe Institute of Technology (KIT) working in the research group of Professor Nestler in the Department of Reliability of Components and Systems. He received his MSc degree in Computer Science from the University of Applied Sciences, Karlsruhe in 2008. As part of the development team for the phase-field code PACE3D he develops algorithms for parallel computation to simulate and study the microstructure behavior of deformed sheet metal during recrystallization in 3D. He is interested in parallel computing with high-performance computers and large-scale 3D microstructure simulations.

Michael Selzer received his Diploma in 2005 and his MSc in 2007 from Hochschule, Karlsruhe and is currently a PhD student at KIT working in the research group of Professor Nestler, where he has an additional role as Department Manager, further developing the phase-field code PACE3D. He has co-authored more than 20 publications and is vastly experienced in numeric optimization, parallel computing, pre/post-processing algorithms and visualization techniques. His current research interests are centered on grain coarsening, fluid flow, sintering processes and elasticity problems.

Johannes Hötzer received his MSc in Computer Science from the University of Applied Sciences, Karlsruhe in 2011 and is currently a PhD student in the research group of Professor Nestler at KIT. The focus of his PhD is the cooperative Promotionskolleg "Gefügestrukturanalyse und Prozessbewertung", the study of sintering phenomena with the phase-field method as well as the parallelization and further development of simulation software. He is interested in sintering processes, modeling large-scale simulations, high-performance computing, and the Linux operating system.

Britta Nestler is the Director of the Institute of Applied Materials at KIT. She is a full Professor of Microstructure Simulations in Materials Sciences. Her group intensively develops simulation methods for modeling phase-transformation problems and pattern formations in multi-component and multiphase material systems. The modeling is based on the phase-field method and combines this approach with other physical evolutions such as fluid flow and continuum mechanics. Before her current position at KIT, she studied Physics and Mathematics at the RWTH Aachen from 1991–1996 and continued her academic career with a PhD at the Foundry Institute at RWTH Aachen. Her PhD was built on a thesis entitled *Phase-Field Modelling of Multiphase Solidification* and finished in 2000. In 2001 she became a Professor at the Karlsruhe University of Applied Sciences. Since then she developed her current research field in computational materials science with a particular focus on microstructure formation on a mesoscopic length scale. She has been honored with a number of prizes such as the Richard von Mises Prize 2002 (GAMM), the Materials Sciences and Technology Prize 2005 (FEMS) and the State Baden–Württemberg Award for Applied Research 2007.