e-Science Journal
Global Scientific Information and Computing Center

Vol. 5
SC'11 Special Issue

TSUBAME ESJ.

SC'11 Special Issue

Gordon Bell Prize

Special Achievements in Scalability and Time-to-Solution
Honorable Mention

Graph500 Ranking No.3

3 Technical Papers

http://www.gsic.titech.ac.jp/TSUBAME_ESJ

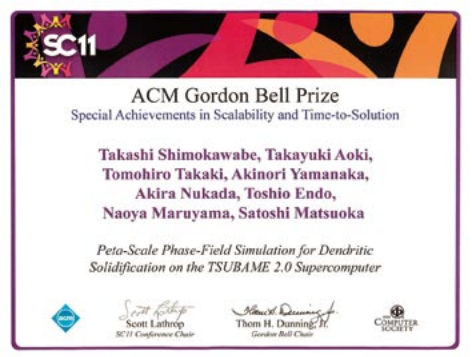
03

ACM Gordon Bell Prize:
Special Achievements in Scalability and Time-to-Solution

& SC'11 Technical Paper

Peta-scale Phase-Field Simulation for Dendritic Solidification on the TSUBAME 2.0 Supercomputer

Takashi Shimokawabe Takayuki Aoki Tomohiro Takaki Akinori Yamanaka
Akira Nukada Toshio Endo Naoya Maruyama Satoshi Matsuoka



SC11
Seattle, WA
Connecting communities through HPC

Peta-scale Phase-Field Simulation for Dendritic Solidification on the TSUBAME 2.0 Supercomputer

Takashi Shimokawabe* Takayuki Aoki** Tomohiro Takaki*** Akinori Yamanaka****
Akira Nukada** Toshio Endo** Naoya Maruyama** Satoshi Matsuoka**

* Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology
** Global Scientific Information and Computing Center, Tokyo Institute of Technology
*** Graduate School of Science and Technology, Kyoto Institute of Technology **** Graduate School of Engineering, Tokyo Institute of Technology

To realize a low-carbon society, it is indispensable to develop new strong and light materials. The material microstructure controlling the mechanical property is determined in the solidification process. The phase-field model is a meso-scale physical theory describing phase transformations such as solidification. Realistic prediction of material solidification requires a millimeter-scale simulation heretofore impossible with supercomputers. The phase-field equation was discretized by using the finite-difference method, and a simulation code in CUDA was developed. A large-scale solidification simulation of an aluminum-silicon alloy was carried out with 4,000 GPUs on TSUBAME 2.0 and achieved 2.0 petaflops in single precision. The ACM Gordon Bell Award - Special Achievement in Scalability and Time-to-Solution was awarded for this achievement in 2011.

1 Introduction

The development of new strong and light materials contributes greatly to transportation systems with high fuel efficiency and realization of a "low-carbon" society. The strength of a metal strongly depends on its microstructure, which is, in turn, determined by the solidification process of the metal (see Fig.1). However, determining material properties (such as strength) requires millimeter-scale macroscopic study.

The phase-field model [1] is a physical theory describing the evolution of complicated material morphologies on the "meso-scale," namely, between the molecular scale and the macroscopic scale of materials. It is a powerful numerical tool for studying the dynamics of phase transformations such as solidification. The derived equations are partial differential equations in time and space and often discretized by FDM (finite-difference method) or FEM (finite element method). The phase-field equations include many complex nonlinear terms, and the amount of computations per mesh or element becomes large compared to normal stencil applications. Furthermore, the phase-field model assumes a narrow-thickness interface between liquid phase and solid phase, so the time integration requires high spatial resolution and short time step, and phase-field simulations remain restricted to two- or small three-dimensional computations due to their large computational cost.

In this study, we carry out a large-scale phase-field simulation for the aluminum-silicon dendritic growth during directional solidification on the GPU supercomputer TSUBAME 2.0. The simulation code is developed in CUDA, and almost all the capability of TSUBAME 2.0 is used for the simulation, which uses a domain decomposition and inter-node communications with an MPI library. Such large-scale phase-field simulation has not been done before and will have a big impact on material science.

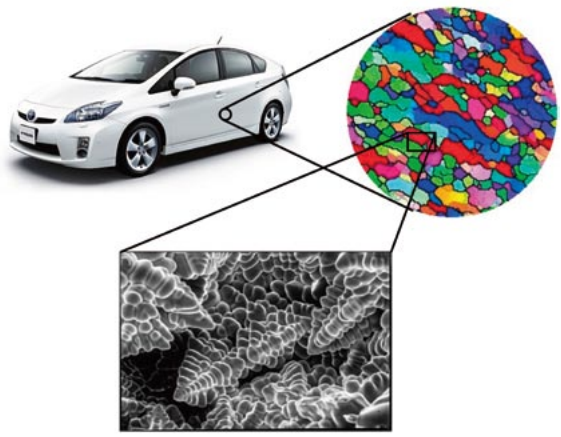


Figure 1 Images of a material microstructure

2 Phase-field model

The phase-field model is derived from non-equilibrium statistical mechanics and can describe meso-scale phenomena that occur on a mid-scale between the molecular and macro scales. The phase-field model introduces a "continuous-order parameter," i.e., a phase-field variable, to describe whether a material is solid or liquid. The phase field ϕ takes a value of 0 in the liquid phase and 1 in the solid phase. Interfaces between solid and liquid phases are treated as diffuse ones described by localized regions, where the parameter changes smoothly between the two fixed values. The interface positions are represented by $\phi = 0.5$. Thanks to this approach, the phase-field method can describe the locations of the interfaces without using a traditional interface-tracking method and carry out the same calculations throughout a computational domain.

The dendritic solidification process in a binary alloy is simulated by solving the phase-field equation and the diffusion

equation of the solute concentration. The time integration of the phase field is described by the following equation, which takes into account the anisotropy of interfacial energy:

$$\frac{\partial \phi}{\partial t} = M_\phi \left[\nabla \cdot (a^2 \nabla \phi) + \frac{\partial}{\partial x}\left(a \frac{\partial a}{\partial \phi_x} |\nabla \phi|^2 \right) + \frac{\partial}{\partial y}\left(a \frac{\partial a}{\partial \phi_y} |\nabla \phi|^2 \right) + \frac{\partial}{\partial z}\left(a \frac{\partial a}{\partial \phi_z} |\nabla \phi|^2 \right) - \Delta S \Delta T \frac{dp(\phi)}{d\phi} - W \frac{dq(\phi)}{d\phi} \right] \quad (1)$$

where a, M_ϕ, W, ΔS, and ΔT represent the gradient coefficient that describes interface anisotropy, the mobility of the phase field, the height of the potential energy barrier, the entropy of fusion and undercooling, respectively. The two functions $p(\phi)$ and $q(\phi)$ are given as $p(\phi) = \phi^3(10 - 15\phi + 6\phi^2)$ and, $q(\phi) = \phi^2(1-\phi)^2$ respectively.

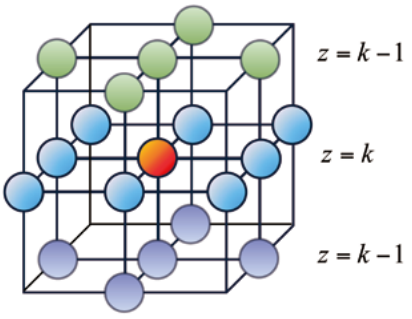
The time integration of the solute concentration is derived by the following diffusion equation:

$$\frac{\partial c}{\partial t} = \nabla \cdot \left[D_S \phi \nabla c_S + D_L (1-\phi) \nabla c_L \right] \quad (2)$$

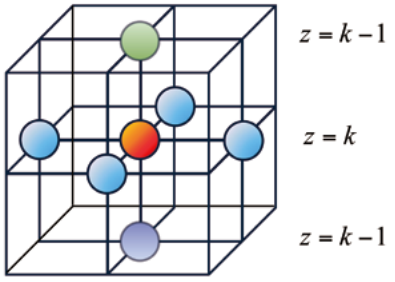
where D_S and D_L are the diffusion coefficients in the bulk solid and liquid phases, respectively. Solid concentration c_S and liquid concentration c_L satisfy $c = (1-\phi)c_L + \phi c_S$.

3 Single-GPU implementation

The time integration of the phase field and the solute concentration given by Equations (1) and (2) are carried out by the second-order finite-difference scheme for space and the first-order forward Euler-type finite-difference method for time on a three-dimensional regular computational grid. The simulation code is written in CUDA to run on GPUs. All variables are allocated on GPU video memory (also called global memory in CUDA), which virtually eliminates all the CPU-GPU data transfers through a PCI-Express bus during simulation runs. The spatial patterns of the neighbor points of phase field ϕ and solute concentration c that are required to solve the discretized governing equations at the center point (i, j, k) of the grid are, respectively, shown in Figures 2 (a) and (b). In one time step , 19 neighbor elements of ϕ and seven neighbor elements of c are used with the governing equations to update the phase field value and concentration value at the center point; thus, 26 elements must be read from the memory and two updated values must be written back to the memory for each point of the grid.



(a) Stencil of phase-field variable.



(b) Stencil of concentration variable.

Figure 2 Spatial access patterns of neighbor points required for time integration.

Peta-scale Phase-Field Simulation for Dendritic Solidification on the TSUBAME 2.0 Supercomputer

The computation in the phase-field model involves a large number of memory accesses. It is thus highly effective to reduce access to the global memory. In GPU computing, a key factor in performance improvement is how threads and blocks are assigned to actual calculation. The elements of the computational domain of size nx × ny × nz are calculated as follows. First, as shown in Figure 3, the given domain is divided into pieces of size 64 × 4 × 32. Our kernel function is invoked on a GPU, and each thread block (which has 64 × 4 × 1 threads) handles each piece. Each thread calculates 32 elements by marching in the z direction; a thread that corresponds to (i, j) updates grid points (i, j, k), where k varies from k0 to k0 + 31, and k0 is a multiple of 32 (i.e., k0 = 0, 32, 64, ..., nz − 32). When a thread computes a point in the k + 1-th plane, some elements to be referred to in this computation have already been accessed by that thread in the previous k-th plane computation. Such data are reused by holding them in registers, thereby reducing global memory accesses. On the other hand, on-chip shared memory is not used to store data shared by several neighbor threads; instead, the L1/L2 cache available on Fermi GPUs including M2050 is relied on.

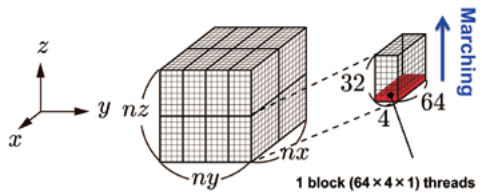


Figure 3 Layouts of CUDA threads and blocks.

4 Optimization of multi-GPU computing

We describe our strategies of multi-GPU computation. Using multiple GPUs is necessary for high-performance computing of large-size problems. In addition to the intrinsic parallel-processing capabilities of GPUs, it is necessary to do parallel computing by using a large number of distributed GPUs. We decompose the whole computational domain in both the y- and z-directions (i.e., 2D decomposition) and allocate each subdomain to one GPU. We have chosen this method because 3D decomposition, which is better in terms of reducing communication amount, tends to degrade GPU performance due to complicated memory access patterns in the y-z plane for data exchanges between the GPU and CPU. Similar to conventional multi-CPU implementations, multi-GPU implementation requires boundary-data exchanges between subdomains. Because a GPU cannot directly access the global memory of other GPUs, host CPUs are used as bridges for data exchange. For inter-node cases, this data exchange is composed of the following three steps: (1) data transfer from the GPU to the CPU by using CUDA APIs, (2) data exchange between nodes with the MPI library, and (3) data transfer back from the CPU to the GPU by using CUDA APIs.

Unlike in conventional multi-CPU computation, in multi-GPU computing, data-communication time with neighbor GPUs is not ignored in the total execution time, especially in large-scale computation, since the performance of the GPU is higher than that of the CPU. To achieve higher performance, it is important to hide communication overheads by overlapping communication with computation. In this study, we implement the following three methods for multi-GPU computing: (a) GPU-only method, (b) Hybrid-YZ method, and (c) Hybrid-Y method.

(a) GPU-only method: This basic method (which is used as a reference) computes all subdomains without applying any specific optimization. After that, the boundary regions of subdomains are exchanged between GPUs by using the above three steps.

(b) Hybrid-YZ method: By dividing each subdomain into five regions, which are two y-boundaries, two z-boundaries and an inside region, we can overlap communication for the boundary data exchange with the computation of the inside region. A similar overlapping technique has been described in previous reports [2]. In this technique, separate GPU kernels are invoked for separate regions. Instead of dividing a GPU kernel, in this study, we let CPU cores compute four y- and z-boundary regions, while the inside

region is computed by GPU. We observed that CPUs often become bottlenecks by taking longer time to do computation of all boundaries and communications than the GPU computation time.

(c) Hybrid-Y method: This method is a slightly modified method from the (b) method. It reduces CPU loads by assigning only the y-boundaries to CPUs instead of both y- and z-boundaries. The GPU can compute the z-boundaries effectively, since it accesses consecutive addresses of the global memory.

Since each node of TSUBAME 2.0 has three GPUs and 12 CPU cores (two 6-core Xeon CPUs), we assign four cores to each of the three GPUs. Each subdomain is thus cooperatively computed by a single GPU and four CPU cores. Multi cores are exploited by using OpenMP.

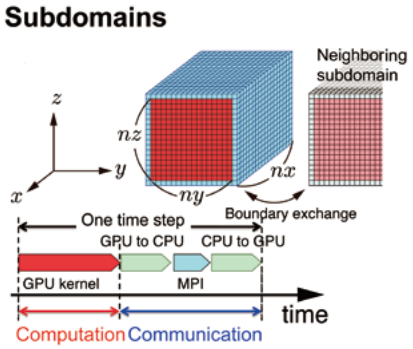


Figure 4 Schematic diagram of the GPU-only method.

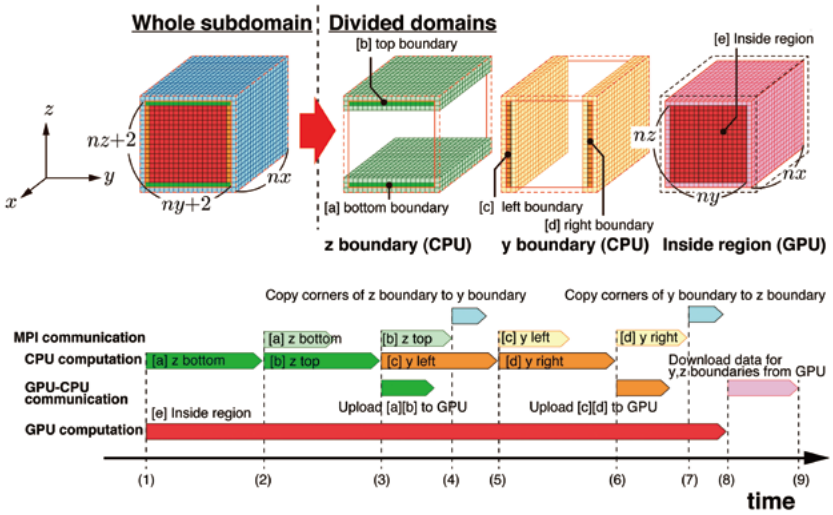


Figure 5 Schematic diagram of the Hybrid-YZ method.

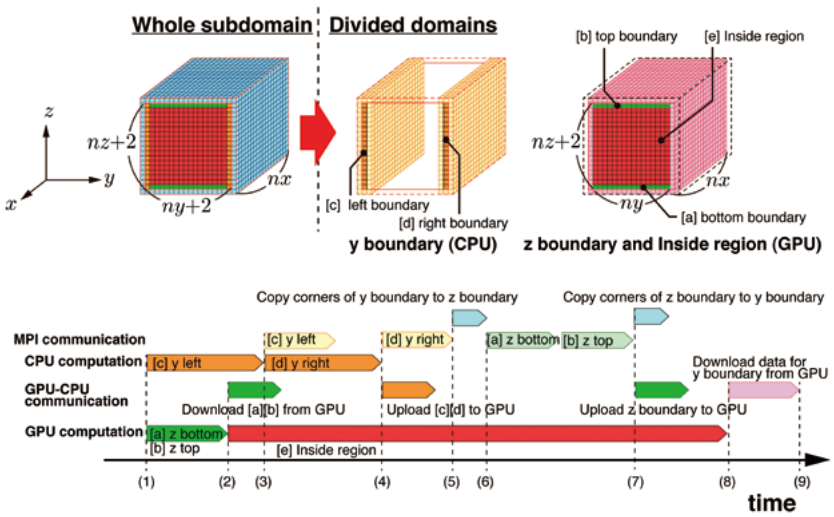


Figure 6 Schematic diagram of the Hybrid-Y method.

Peta-scale Phase-Field Simulation for Dendritic Solidification on the TSUBAME 2.0 Supercomputer

Performance of multi-GPU computing 5

We show the performance of (a) the GPU-only method, (b) the Hybrid-YZ method, and (c) the Hybrid-Y method, all using multiple M2050 GPUs of the GPU-rich supercomputer TSUBAME 2.0 at the Global Scientific Information and Computing Center of the Tokyo Institute of Technology. In this study, we simulate solidification of aluminum–silicon (Al–Si) alloy.

An image of the solidification process of the alloy observed at the world's largest synchrotron radiation facility, SPring- 8, by Hideyuki Yasuda, a professor at Osaka University, and his research group, is shown in Figure 7 (a) . Results of the phase-field simulation using a 4096 × 128 × 4096 mesh are shown in Figure 7 (b) . Although the alloy systems in the images differ, the growth process obtained by the simulation agrees well with the observed one.

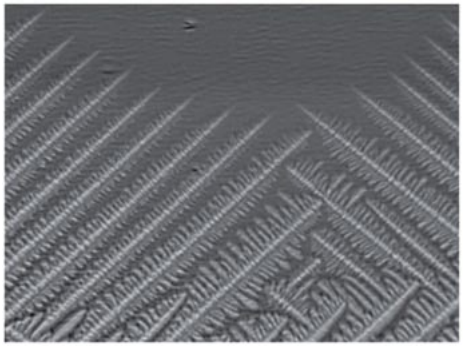


Figure 7(a) Solidification process of an alloy observed at SPring-8 (courtesy of Professor Yasuda).

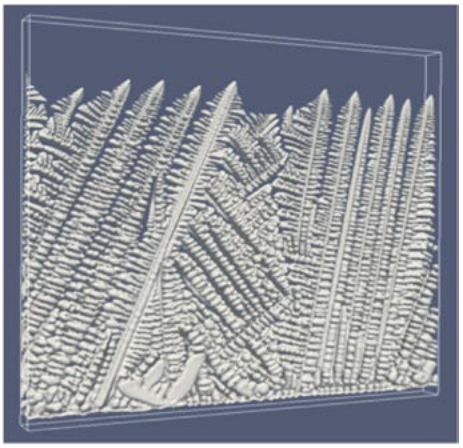


Figure 7(b) Solidification growth simulated by the phase-field model using GPUs.

To measure the performance of the GPU computation, we count the number of floating-point operations in the C/C++-based phase-field simulation code by running it on a CPU with a performance counter provided by PAPI (Performance API). The obtained count and GPU elapsed time are used for evaluating the performance of the GPU computing. Figure 8 shows the results on strong scaling, which show variation of the performance with the number of GPUs for a fixed total mesh size. This figure compares the strong-scaling characteristics of the three proposed methods i.e., (a), (b), and (c). We perform simulations in single precision for three different mesh sizes: 512^3, 1024^3, and 2048^3. It is clear from the figure that the overlapping methods, (b) Hhybrid-YZ method and (c) Hhybrid-Y method, both work effectively in hiding communication overhead as expected, resulting in an improvement of overall performance when a small number of GPUs are used. When the number of GPUs is larger, the volume that each GPU handles becomes smaller, and the percentage of the boundary region computed by the CPU in the whole computational domain increases. Eventually, the GPU computation is no longer able to hide the communication cost. In the case of the Hybrid-YZ method, method (b), we observed that the CPUs often become bottlenecks by taking a longer time to do their operations than the GPU computation time especially when the number of GPUs is larger. Since the Hybrid-Y method alleviates this bottleneck, it achieves a significantly improved performance compared with that of the GPU-only method.

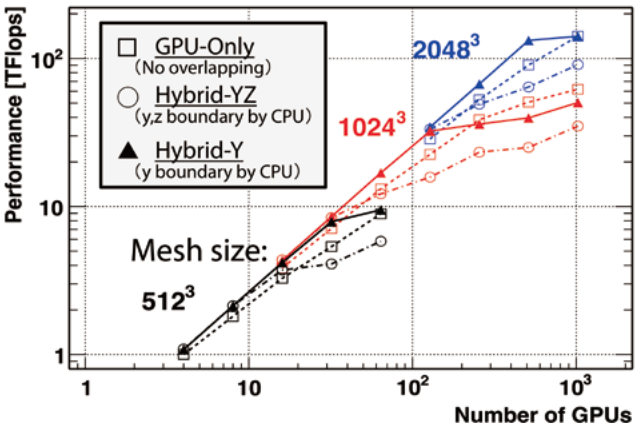


Figure 8 Results for strong scaling of multi-GPU computation in single precision.

Next, we show the weak scaling results, which show how the performance of the three methods varies with the number of GPUs for a fixed mesh size per GPU. Each GPU handles a domain of 4096 × 160 × 128 for single precision. In demonstrating weak

scalability, the Hybrid-Y method has achieved extremely high performance in single precision (Figure 9), namely, reaching 2.000 petaflops, specifically 1.975 petaflops by the GPU and 0.025 petaflops by CPU, for a 4096 × 6480 × 13000 mesh using 4,000 GPUs along with 16,000 CPU cores. This is the first peta-scale result as a real stencil application we know to date.

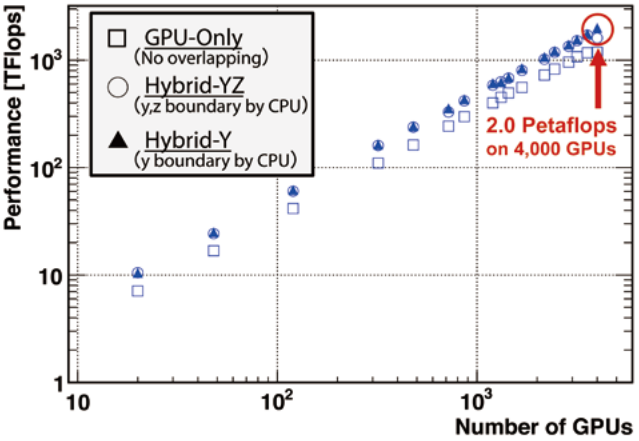


Figure 9 Results on weak scaling of multi-GPU computation in single precision.

Figure 10 shows that electric power consumption of TSUBAME 2.0 when the simulation achieved 2 petaflops, namely, 44.5% of the efficiency to the peak performance, using 4,000 GPUs along with 16,000 CPU cores. The 2-petaflops simulation consumed electric power of 1.36 MW for all computational nodes and networks on TSUBAME 2.0. The simulation thus achieved 1468 Mflops/W. These results show that the simulation results were obtained by small electric power consumption.

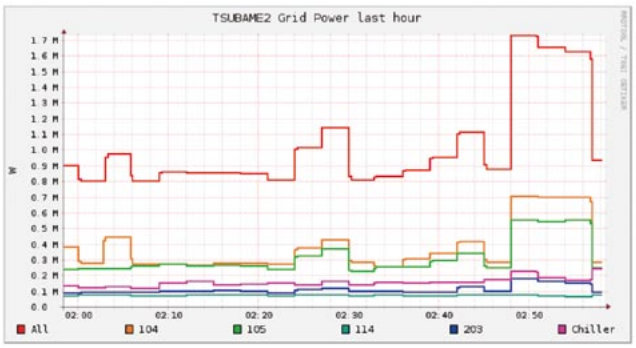


Figure 10 Electrical power consumption of the 2-petaflops phase-field simulation running on TSUBAME 2.0.

Figure 11 demonstrates the dendritic growth during binary-alloy solidification using a mesh size of 4096 × 1024 × 4096 on TSUBAME 2.0. As an initial condition, 32 nuclei are put on the z = 0 plane. This simulation will allow us to clarify the mechanisms of competitive growth between parallel dendrites. Evaluating these mechanisms is important to design and control the mechanical properties of solidified products.

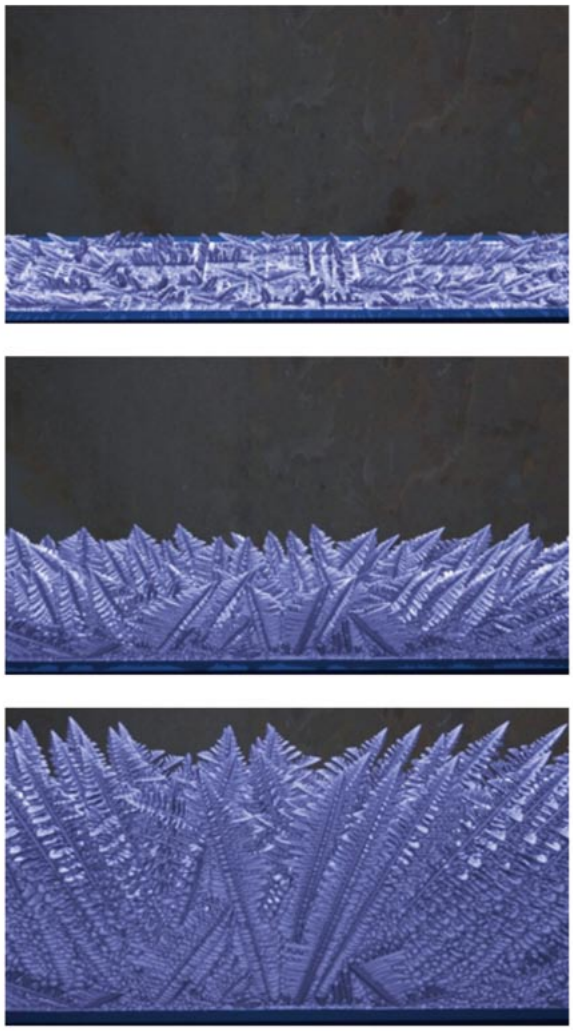


Figure 11 Large-scale simulation of dendritic solidification of aluminum–silicon alloy.

Peta-scale Phase-Field Simulation for Dendritic Solidification on the TSUBAME 2.0 Supercomputer

Summary

A large-scale GPU simulation of the dendritic growth during binary-alloy solidification was carried out on the basis of a phase-field model. An extremely high performance of 2.0 petaflops in single precision was achieved on 4,000 GPUs of TSUBAME 2.0, in spite of a stencil application, which is hard to extract high performance. The computation reached 44.5% of the peak performance and simultaneously high efficiency from the viewpoint of electrical power consumption. It means that we have the computational result on TSUBAME 2.0 with smaller energy compared to conventional supercomputing. It is concluded that a GPU supercomputer is available for various practical stencil applications.

Acknowledgements

The peta-scale simulation was executed as a TSUBAME Grand Challenge Program 2011 spring and we express our special thanks to have a chance to use the whole TSUBAME resources. This research was supported in part by Grant-in-Aid for Scientific Research (B) 23360046 from The Ministry of Education, Culture, Sports, Science and Technology (MEXT), two CREST projects, "ULP-HPC: Ultra Low-Power, High Performance Computing via Modeling and Optimization of Next Generation HPC Technologies" and "Highly Productive, High Performance Application Frameworks for Post Petascale Computing" from Japan Science and Technology Agency (JST), and JSPS Global COE program "Computationism as a Foundation for the Sciences" from Japan Society for the Promotion of Science (JSPS).

References

[1] R. Kobayashi: Modeling and numerical simulations of dendritic crystal growth. Physica D, Nonlinear Phenomena, 63(3-4), 410 - 423 (1993)

[2] Takayuki Aoki, Satoi Ogawa, Akinori Yamanaka: GPU Computing for Dendritic Solidification based on Phase-Field Model, TSUBAME e-Science Journal, Vol.1, pp.5-8, Global Scientific Information and Computing Center, Tokyo Tech (2010 September)

[3] A. Yamanaka, T. Aoki, S. Ogawa, and T. Takaki: GPU-accelerated phase-field simulation of dendritic solidification in a binary alloy. Journal of Crystal Growth, 318(1):40 - 45 (2011). The 16th International Conference on Crystal Growth (ICCG16)/ The 14th International Conference on Vapor Growth and Epitaxy (ICVGE14)

[4] T. Shimokawabe, T. Aoki, T. Takaki, A. Yamanaka, A. Nukada, T. Endo, N. Maruyama, and S. Matsuoka : Peta-scale Phase-Field Simulation for Dendritic Solidification on the TSUBAME 2.0 Supercomputer, in Proceedings of the 2011 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC'11, IEEE Computer Society, Seattle, WA, USA, Nov. 2011.