Progress in NUCLEAR SCIENCE and TECHNOLOGY, Vol. 2, pp.639-642 (2011)

ARTICLE

Multiple-GPU Scalability of Phase-Field Simulation for Dendritic Solidification

Takayuki AOKI[1,*], Satoi OGAWA[1] and Akinori YAMANAKA[2]

[1] *Tokyo Institute of Technology, 2-12-1 O-okayama, Meguro-ku, Tokyo 152-8550, Japan*
[2] *Graduate School of Science and Engineering, Tokyo Institute of Technology*

Mechanical properties of metallic materials like steel depend on the solidification process. In order to study the morphology of the microstructure in the materials, the phase-field model derived from the non-equilibrium statistical physics is applied and the interface dynamics is solved by GPU computing. Since very high performance is required, 3-dimensional simulations have not been carried out so much on conventional supercomputers. By using 60 GPUs installed on TSUBAME 1.2, a performance of 10 TFlops is archived for the dendritic solidification based on the phase-field model.

KEYWORDS: GPGPU, phase-field model, dendritic solidification, CUDA, mupltiple GPU, TSUBAME 1.2, Tesla C1060

I. Introduction

The mechanical properties of metallic materials are strongly characterized by distribution and morphology of the microstructure in the materials. In order to improve the mechanical performance of the materials and to develop a new material, it is essential to understand the microstructure evolution during solidification and phase transformation. Recently, the phase-field model[1)] has been developed as a powerful method to simulate the microstructure evolution. In the phase-field modeling, the time-dependent Ginzbunrg-Landau type equations which describe interface dynamics and solute diffusion during the microstructure evolution are solved by the finite difference and finite element methods. This microstructure modeling has been applied to numerical simulations for solidification, phase transformation and precipitation in various materials. However, large computational cost is required to perform realistic and quantitative three-dimensional phase-field simulation in the typical scales of the microstructure pattern. To overcome such computational task, we utilize the GPGPU (General-Purpose Graphics Processing Unit)[2)] which is developed as an innovative accelerator[3)] in high performance computing (HPC) technology.

GPU is a special processor often used in personal computers (PC) to render the graphics on the display. The request for high-quality computer graphics and PC games led great progress on GPU's performance and made it possible to apply it to general-purpose computation. Since it is quite different from the former accelerators due not only to high performance of floating point calculation but also a wide memory bandwidth, GPGPU is applicable to various types of HPC applications. In 2006, CUDA[3)] framework was released by NVIDIA and it has enabled us GPU programming in standard C language without taking account for graphics functions.

In this article, we study the growth of the dendritic solidification of a pure metal in super cooling state by solving the equations derived from the phase-field model. Finite difference discretization is employed and the GPU code developed in CUDA is executed on the GPU of TSUBAME 1.2. The remarkably high performance is shown in comparison with the conventional CPU computing. Although most GPGPU applications run on single GPU, we exploit a multiple GPU code and show the strong scalability of large-scale problems.

II. Phase-Field Model

The phase-field model is a phenomenological simulation method to describe the microstructure evolution in sub-micron scale based on the diffuse-interface concept. In this article, to describe the interface between solid phase and liquid phase, we define a non-conserved order parameter (phase field) ϕ taking a value of 0 in the liquid phase and 1 in the solid phase. In the interface region, ϕ gradually changes from 0 to 1. The position at $\phi = 0.5$ can be defined as the solid/liquid interface. Using this diffuse-interface approach, the phase-field method does not require explicit tracking of the moving interface.

In this study, a time-dependent Ginzbunrg-Landau equation for the phase field ϕ and a heat conduction equation are solved.[4)] The governing equations for the phase field ϕ and the temperature T are given by the following equations.

$$\frac{\partial \phi}{\partial t} = M \left[\begin{array}{l} \frac{\partial}{\partial x}\left(\varepsilon^2 \frac{\partial \phi}{\partial x} + \varepsilon \frac{\partial \varepsilon}{\partial \phi_x} |\nabla \phi|^2 \right) + \frac{\partial}{\partial y}\left(\varepsilon^2 \frac{\partial \phi}{\partial y} + \varepsilon \frac{\partial \varepsilon}{\partial \phi_y} |\nabla \phi|^2 \right) \\ + \frac{\partial}{\partial z}\left(\varepsilon^2 \frac{\partial \phi}{\partial z} + \varepsilon \frac{\partial \varepsilon}{\partial \phi_z} |\nabla \phi|^2 \right) + 4W\phi(1-\phi)\left\{ \phi - \frac{1}{2} + \beta + a\chi \right\} \end{array} \right] \quad (1)$$

*Corresponding author, E-mail: taoki@gsic.titech.ac.jp

$$\frac{\partial T}{\partial t} = \kappa \nabla^2 T + 30\phi^2(1-\phi)^2 \frac{L}{C}\frac{\partial \phi}{\partial t} \tag{2}$$

Here, M is the mobility of the phase filed ϕ, ε is the gradient coefficient, W is the potential height and β is the driving force term. These parameters are related to the material parameters as follows.

$$M = \frac{bT_m\mu}{3\delta L}, \tag{3}$$

$$W = \frac{6\sigma b}{\delta}, \tag{4}$$

$$\varepsilon = \sqrt{\frac{3\delta\sigma}{b}}\left(1 - 3\gamma + 4\gamma\frac{\phi_{,x}^4 + \phi_{,y}^4 + \phi_{,z}^4}{|\nabla\phi|^4}\right), \tag{5}$$

$$\beta = -\frac{15L}{2W}\frac{T - T_m}{T_m}\phi(1-\phi)\cdot \tag{6}$$

In this article, the material parameters for pure nickel are used. The melting temperature T_m = 1728 K, the kinetic coefficient μ = 2.0 m/Ks, the interface thickness δ = 0.08 μm, the latent heat L = 2.35 × 10^9 J/m^3 and the interfacial energy σ = 0.37 J/m^2, the heat conduction coefficient κ = 1.55 × 10^{-5} m^2/s and the specific heat C = 5.42 × 10^6 J/Km^3. Furthermore, the strength of interfacial anisotropy γ = 0.04. Assuming the interface region $\lambda < \phi < 1-\lambda$, we obtained $b = \tanh^{-1}(1-2\lambda)$. λ is set to be 0.1, so that b reduce to 2.20. χ is a random number distributed uniformly in the interval [−1, 1]. a is the amplitude of the fluctuation and set to be 0.4.

III. GPU Computing

The calculations in this article were carried out on the TSUBAME Grid Cluster 1.2 in the Global Scientific Information and Computing Center (GSIC), at Tokyo Institute of Technology. Each node consists of the Sun Fire X4600 (AMD Opteron 2.4 GHz 16 cores, 32 GByte DDR2) and is connected by two Infiniband SDR networks with 10 Gbps. Two GPUs of the NVIDIA Tesla S1070 (4 GPUs, 1.44GHz, VRAM 4GByte, 1036 GFlops, memory bandwidth 102 GByte/s) are attached to 340 nodes (total 680 GPUs) through the PCI-Express bus (Generation 1.0 x8). In our computations, one of the two GPUs is used per a node. The Opteron CPU (2.4 GHz) on each node has the performance of 4.8 GFlops and a memory bandwidth of 6.4 GByte/sec (DDR-400). CUDA version 2.2 and NVIDIA Kernel Module 185.18.14 are installed and the OS is SUSE Enterprise Linux 10.

1. Tuning Techniques

Equations (1) and (2) are discretized by the second-order Finite Difference Method and time-integrated with the first-order accuracy (Euler scheme).The arrays for the dependent variables ϕ at the n and n+1 time steps are allocated on the VRAM, called the global memory in CUDA. We minimize the data transfer between the host (CPU) memory and the device memory (global memory) through the narrow PCI-Express bus, which becomes a large overhead of the GPU computing.

In CUDA programming, the computational domain of a $nx \times ny \times nz$ mesh is divided into $L \times M \times M$ smaller domains of a $MX \times MY \times MZ$ mesh, where $MX = nx/L$, $MY = ny/M$, $MZ = nz/N$. We assign the CUDA threads (MX, MY, 1) to each small domain in the x- and y-directions. Each thread computes MZ grid points in the z-direction using a loop. The GPU performance strongly depends on the block size and we optimize it to be MX = 64 and MY = 4.

The discretized equation for the phase field variable ϕ is reduced to the stencil calculation referring to 18 neighbor mesh points. In order to suppress the global memory access, we use the shared memory as a software managed cache. Recycling three arrays with a size of (MX + 2)×(MY + 2) saves the use of the shared memory. In computing the temperature, the shared memory is used similarly, however in this case the time derivative term $\partial\phi/\partial t|_{i,j,k}^{n}$ appears in the right-hand side of Eq.(2). We fuse the computational kernel function of $\phi_{i,j,k}^{n} \to \phi_{i,j,k}^{n+1}$ with the kernel function of $T_{i,j,k}^{n} \to T_{i,j,k}^{n+1}$, so that it becomes unnecessary to access the global memory by keeping the value $\partial\phi/\partial t|_{i,j,k}^{n}$ on a temporal variable in the kernel function.

2. Performance of Single GPU

In order to evaluate the performance of the GPU computing and check the numerical results in comparison with CPU, we also built the CPU code simultaneously. Since integer calculations are also done by streaming processors of GPU, we count the number of floating point operation of the calculation for the dendrite solidification by using the hardware counter of the PAPI (Performance API)[5)] for the CPU code.

The maximum mesh size of the run is 640×640×640 on single GPU, because one GPU board of Tesla S1070 has 4 GByte VRAM (GDDR3). By changing the mesh size, we measured the performance of the GPU computing and we

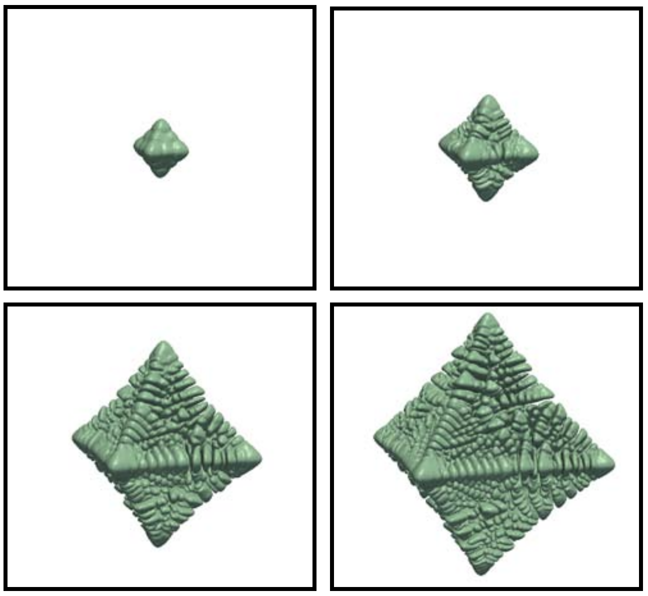

Fig. 1 Snap shots of the dendritic solidification growth

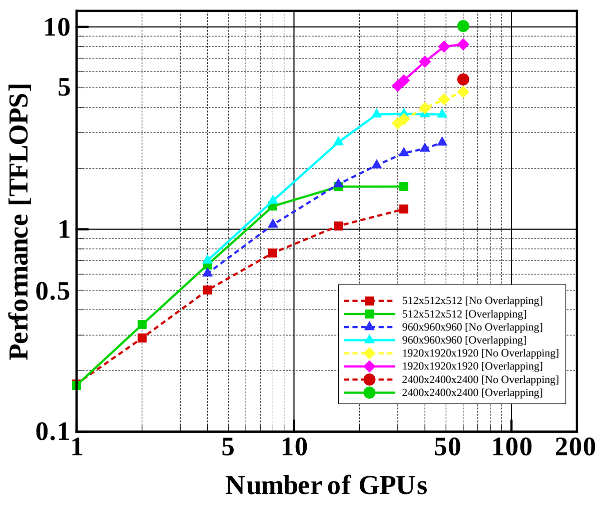

Fig. 2 Strong Scalabilities of multi-GPU computing

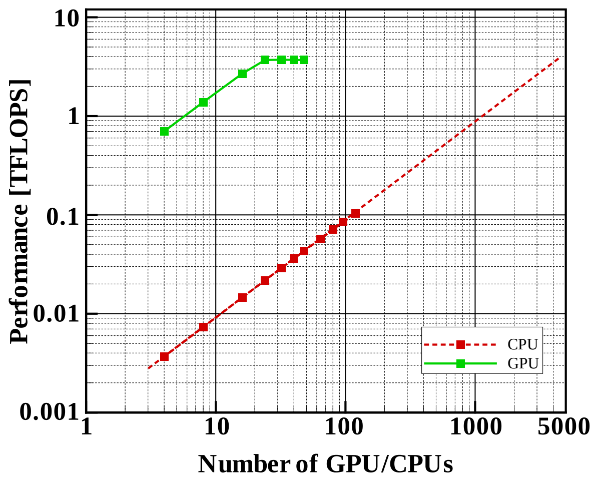

Fig. 3 Performance Comparison between GPU and CPU on TSUBAME 1.2 for the case of 960×960×960 mesh

had 116.8 GFlops for 64×64×64 mesh, 161.6 GFlops for 128×128×128 mesh, 169.8 GFlops for 256×256×256 mesh, 168.5 GFlops for 512×512×512 mesh and 171.4 GFlops 640×640×640 mesh. The performance of the single CPU core (Opteron 2.4 GHz) is 898 MFlops, and it was found to be a 190x-speedup on TSUBAME1.2.

The phase-field calculation consists of 373 floating point operations and 28 times global memory access (26 reads and 2 writes) per one mesh point. The same calculation is carried out on every mesh point in single precision. The arithmetic intensity is estimated to be 3.33 Flop/Byte. In the case when using the shared memory, the number of memory read reduces to 2 and the arithmetic intensity increases up to 23.31 Flop/Byte. It is understood that the calculation is much more compute-intensive than Computational Fluid Dynamics standards. Therefore, such high performances as 171.4 GFlops can be achieved in the GPU computing. **Figure 1** shows the snap shots of the dendritic solidification growth.

IV. Multiple-GPU Computing

1. GPU Computing on Multi-Node

Multiple GPU computing is carried out for the following two purposes: (1) enabling large-scale computing beyond the memory limitation on a single GPU card and (2) speedup the fixed problem pursuing strong scalability. Multiple GPU computing requires GPU-level parallelization constructing a hierarchical parallel computing, since the blocks and the threads in CUDA have already been parallelized inside the GPU. The computational domain is decomposed and a sub-domain is assigned to each GPU.

Using the MPI library for the communication between GPU nodes, we run the same process number as the GPU number. Direct data transfer of GPU-to-GPU is not available and a three-hop communication is required: global memory to host memory, MPI communication, host memory to the global memory. This communication overhead of the multi-GPU computing is relatively much larger than that of CPU. In this article a 1-dimensional domain decomposition is examined for simplicity.

2. Overlap between Communication and Computation

In order to improve the performance of the multiple-GPU computing, an overlapping technique between communication and computation is introduced. Since Eqs. (1) and (2) are explicitly time-integrated, only the data of one mesh layer at the sub-domain boundary is transferred. The GPU kernel is divided into two and the first kernel computes the boundary mesh points. At this moment, the data is ready to be transferred and the CUDA memory copy API from the global memory to host memory starts asynchronously as the stream 0. Simultaneously the second kernel that computes the inner mesh points starts as the stream 1. The overlapping of stream 0 with the stream 1 can hide the communication time.

3. Performance of Multiple GPU Computing

In the four cases: 512×512×512 mesh, 960×960×960 mesh, 1920×1920×1920 mesh and 2400×2400×2400 mesh, their performances are examined with changing the number of GPUs for both the overlapping and the non-overlapping cases. The strong scalabilities are shown in **Fig.2**. All the cases were done by using one of two GPUs per a node combined and connected by single interface card, since we want to maintain the GPU-to-GPU communication bandwidth as much as possible.

In every case, the performance of the overlapping computation is greatly improved compared with that of the non-overlapping. The ideal strong scalabilities are achieved up to 8 GPUs for 512×512×512 mesh, from 4 to 24 GPUs for 960×960×960 mesh, from 30 to 48 GPUs for 1920×1920×1920 mesh. We have a perfect weak scalability in the extent of the GPU number used in our runs.

In the overlapping cases, the ideal strong scalabilities are suddenly saturated by increasing the GPU number. In one-dimensional domain decomposition, the communication time does not depend on the GPU number. On the other hand, the domain size decreases with increasing the GPU number. For instance, runs of 512×512×512 have the longer computational time than the communication with less than 32 GPUs, however the computational time becomes shorter than the communication for greater than 32 GPUs. It is not possible to hide the communication time any more.

It should be highlighted that the performance of a 10 TFlops is achieved with 60 GPUs, which is comparable performance of the application running on world top-class supercomputers. It took about 1 hour of the elapsed time to achieve the full growth in the domain as shown in the last snap of Fig. 1. We directly compare the GPU performance with the CPU on TSUBAME 1.2 for the same test case of 960×960×960 mesh. The CPU code is not tuned by using SSE and only the optimization option of the compiler is applied. In the overlapping case, 24 GPUs show the performance of 3.7 TFlops in **Fig. 3**, and it is noticed that 24 GPUs are comparable with 4000 Opteron (2.4 GHz) CPU cores, even if we assume the perfect strong scalability.

V. Conclusion

The GPU computing for the dendritic solidification process of a pure metal was carried out on the NVIDIA Tesla S1070 GPUs of TSUBAME 1.2 by solving a time-dependent Ginzbunrg-Landau equation coupling with the thermal conduction equation based on the phase-field model. The GPU code was developed in CUDA and a performance of 171 GFlops was achieved on a single GPU. It is found that the multiple-GPU computing with domain decomposition has a large communication overhead. Both the strong and the weak scalabilities were shown. A performance of 10 TFlops was achieved with 60 GPUs, when the overlapping technique was introduced. The GPU computing greatly contributes to low electric-power consumption and is a promising candidate for the next-generation supercomputing.

Acknowledgment

This research was supported in part by KAKENHI, Grant-in-Aid for Scientific Research(B) 19360043 from The Ministry of Education, Culture, Sports, Science and Technology (MEXT), JST-CREST project ”ULP-HPC: Ultra Low-Power, High Performance Computing via Modeling and Optimization of Next Generation HPC Technologies” 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) T. Takaki, T. Fukuoka, Y. Tomita, “Phase-field simulation during directional solidification of a binary alloy using adaptive finite element method,” *J. Crystal Growth*., **283**, 263-278 (2005).

2) T. Endo, S. Matsuoka, “Massive Supercomputing Coping with Heterogeneity of Modern Accelerators,” *IEEE International Parallel & Distributed Processing Symposium (IPDPS 2008)*, (2008).

3) NVIDIA Corporation, *NVIDIA CUDA Compute Unified Device Architecture Programming Guide Version 2.0*, NVIDIA Corporation, California, (2008).

4) R. Kobayashi, “Modeling and numerical simulations of dendritic crystal growth,” *Physica*, **D63**[3-4], 410-423 (1993).

5) PAPI, http://icl.cs.utk.edu/papi/