Parallel Implementation for Phase-Field Simulation of Flow Effect on Dendritic Growth with GPU Acceleration

Changsheng Zhu1,2,*, Jinfang Jia1, Hong Zhang1, Rongzhen Xiao2 and Li Feng2

1College of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China
2State Key Laboratory of New Non-ferrous Metal Materials, Lanzhou University of Technology, Lanzhou 730050, Gansu, China

A Sola-phase field model combined Sola algorithm with phase-field model is established. It is difficult to implement real-time simulation as the computational grids increase. Taking pure SCN for example, the solidification microstructure evolution process in the presence of flow has been accelerated on a GPU with CUDA programming. The GPU implementation of the Sola-phase field model is introduced in this paper. The acceleration results of the dendritic growth simulation under flow by using a single NVIDIA GeForce GTX780 GPU with different memories are also evaluated. The results show that the GPU computation with the shared memory achieves the best acceleration effect, which is 56.16 times faster than that on a single CPU core for 2048 × 2048 grid size. In addition, the simulation results on GPU tally well with that on CPU, which indicates the reliability of GPU-accelerated phase-field simulation. [doi:10.2320/matertrans.M2014269]

(Received July 24, 2014; Accepted August 26, 2014; Published November 25, 2014)

Keywords: phase-field model, Sola algorithm, flow, dendritic growth, graphic processing unit (GPU)

1. Introduction

From a macro point of view, crystal growth is actually a transportation process of heat, mass and momentum. The transportation process plays a limited role in crystal growth rate and affects the growth interface stability.1) There exists a natural convection caused by temperature difference and concentration difference besides the momentum convection in melt convection during solidification process. The natural convection is always present in the melt solidification process, which has an important influence on the structure and composition segregation after solidification,2) and directly affects the solidification defect distributions of solute segregation, solidification porosity and cavity. Therefore, the study on the flow effect during microstructure solidification is of great significance.

In recent years, research on microstructure formation rule under convection using computer simulation technology is one of the research focuses and frontier topics in the microstructure solidification research field. Traditional dendritic growth simulation based on sharp interface model must accurately track complex solid-liquid interface, and the calculation is difficult. However, phase-field method not only avoids the real-time tracking for boundary, but does not need to judge the explicit boundary conditions repeatedly. Thus, the phase-field method has become a computing tool used to simulate complex graphics causes in the crystal growth process.3–5) With the development of computational science, the melt convection is gradually coupled into microstructure simulations.6–11) As phase-field method is applied more and more widely in solidification microstructure simulation, the established phase-field model is more and more complicated. Large-scale simulation is required to realize the actual casting microstructure simulation, which makes the calculation amount is a prominent bottleneck by using finite difference method. So, adopting reasonable and efficient numerical solution method for phase-field model is very important.

Some studies in this respect have been carried out by scholars. LIN et al.12) simulated the three-dimensional phase-field model of dendritic crystal growth with high anisotropy using an adaptive mesh method. DHOTE et al.13) implemented the phase-field models for shape memory alloys using distributed computing, and the numerical implementation is based on the isogeometric analysis framework. JELINEK et al.14) simulated an extremely lattice Boltzmann (LB)-cellular automation (CA) model of two-dimensional dendritic solidification under forced convection using MPI technique. GUO et al.15) studied alloy dendrite growth during solidification with coupled thermal-solute-convection fields using a novel parallel-multigrid numerical approach. Yamanaka et al.16) implemented the three-dimensional phase-field simulation for dendritic solidification of a binary alloy on a single NVIDIA Tesla C1060 GPU, the simulation results showed that the GPU calculation for 5763 computational grids is 100 times faster than that with a single CPU core under the same conditions. Takaki et al.17) performed very-large-scale phase-field computations by a GPU supercomputer, and showed the survival of unfavorably oriented dendrites and highly complicated dendrite-dendrite interactions in three-dimensional space during the directional solidification of a binary alloy.

In this paper, the correctional Sola algorithm using adaptive pressure iteration method combine with the phase-field model to establish a Sola-phase field model of undercooled melt dendrite growth under flow. In the numerical simulation, it is time-consuming to iteratively solve the pressure field, and difficult to meet real-time requirements when the computational scale increases. Hence, study on how to improve the speed of solving the pressure equation becomes a very important issue.18) For phase-field model, due to the adoption of uniform grids and the equations with good parallelism, it is suitable for using the GPU and CUDA to realize parallel solution to phase-field simulation, with the purpose of shortening computation time and improving computational efficiency.
2. Phase-Field Model

2.1 Phase-field equation and temperature field equation

The phase-field and temperature field controlling equations coupling with thermal noise can be defined as

\[
\tau(\vec{n})\partial_t \phi = [\phi - \lambda u(1 - \phi^2)](1 - \phi^2) + V|W^2(\vec{n})|\nabla \phi
\]

\[+
\sum_{m=x,y} \partial_m \left[ |\nabla \phi|^2 W(\vec{n}) \frac{\partial W(\vec{n})}{\partial \partial_m \phi} \right]
\]

(1)

\[
\partial_t u + (1 - \psi)V\nabla u = D\nabla^2 u + \frac{1}{2} \bar{\rho} \phi - \nabla \cdot \bar{q}(\vec{r}, t)
\]

(2)

\[|V| = \sqrt{V_0^2 + V_y^2}
\]

(3)

where \(\phi\) is the phase field variable; \(\lambda\) is coupling constant. The dimensionless temperature \(u\) is defined as \(u = (T - T_M)/(L/C_p)\), where \(T_M, L\) and \(C_p\) are the melting temperature, latent heat and specific heat, respectively. \(\psi\) can be viewed as a solid fraction: \(\psi = (1 + \phi)/2, \psi \in [0, 1]\). \(V_0, V_y\) and \(V_0^2\) are the dimensional variables of melt inflow velocity along directions \(x\) and \(y\), respectively. Their dimensionless forms are: \(V_0, V_y\) and \(V_0^2\) are the dimensionless variables of melt inflow velocity along directions \(x\) and \(y\), respectively. Their dimensionless forms are: \(V_0, V_y\) and \(V_0^2\).

\[
\frac{\partial}{\partial t} \frac{\bar{\rho} \phi}{\bar{\rho}_0} = D_\phi \nabla^2 \phi + V_0 \nabla \phi + \frac{1}{2} \bar{\rho} \phi - \nabla \cdot \bar{q}(\vec{r}, t)
\]

(4)

\[
W(\vec{n}) = W_0(1 - 3\varepsilon_4) \left[ 1 + \frac{4\varepsilon_4}{1 - 3\varepsilon_4} \left( \frac{\partial \phi}{\partial \vec{n}} \right)^2 \right]
\]

(5)

where \(\varepsilon_4\) is the anisotropy strength of the surface energy.

\[
q_m(\vec{r}, t)q_n(\vec{r}', t') = 2DK_0 C_p T_0^2 \delta(\vec{r} - \vec{r}') \delta(t - t')
\]

(6)

where \(\vec{r}\) and \(\vec{r}'\) are position vectors; \(t\) and \(t'\) are time variables. \(K_0\) is the Boltzmann constant; \(\delta(\cdot)\) is Kronecker function; \(\delta\) is the delta function. Some variables can be converted into their dimensionless forms as: \(\vec{r}/W_0 \rightarrow \vec{r}\), \(t/T_0^2 \rightarrow t\), \(\vec{r}/W_0^2 \rightarrow D,\ \vec{q},\ \tau_0/W_0 \rightarrow \vec{q},\ \vec{W}_0\) and \(\tau_0\) are the length scale and time scale, and then the thermal noise vectors are given by

\[
q_m(\vec{r}, t)q_n(\vec{r}', t) = 2DF_w \delta(\vec{r} - \vec{r}') \delta(t - t')
\]

(7)

where the thermal noise amplitude \(F_w\) is defined as

\[
F_w = \frac{k_B T_0^2 C_p}{L^2 W_0^2} = \frac{k_B T_0^2 C_p}{L^2 W_0^2} \left( \frac{d_0}{W_0} \right)^2 = \frac{F_{\text{exp}}}{\left( \frac{d_0}{W_0} \right)^2}
\]

(8)

2.2 Mass and momentum conservation equations

The mass conservation equation and momentum equation are coupled to the phase-field equation in the following way. To describe the liquid melt flow between liquid phase region and dendrite, assuming that the flow in solidification process does not cause the grain position change.\(^1\)

The mass conservation equation is given by

\[
\nabla \cdot ((1 - \phi) \vec{V}) = 0
\]

(9)

The momentum conservation equation can be defined as

\[
\frac{\partial}{\partial t} \frac{\bar{\rho} \vec{V}}{\bar{\rho}_0} + \nabla \cdot \frac{\bar{\rho} \vec{V}}{\bar{\rho}_0} = -\frac{1}{\bar{\rho}} (\vec{V} \nabla \rho + \nabla \mu \nabla (1 - \phi) \vec{V}) - m\vec{V}
\]

(10)

where \(\rho, \mu\) and \(P\) are the density, kinematic viscosity and dimensionless pressure, respectively. Here, assuming the solid and liquid phase densities are the same. The term \(m\) stands for interfacial drag coefficient and is given by

\[
m = \frac{\mu \phi^2 (1 - \phi)}{W^2}
\]

(11)

where \(h\) is the dimensionless constant between 2.55~2.757 related to the interface thickness.

3. Initial Condition and Simulation Parameters

For an initial nucleus of the radius \(r_0\), all the initial conditions are shown as follows:

\[
\begin{align*}
\phi &= 1 & u &= 0 & V_x &= 0 \\
V_y &= 0 & P &= 0 & x^2 + y^2 &\leq r_0^2 \\
\phi &= -1 & u &= -\Delta & V_x &= 0 \\
V_y &= U & P &= 0 & x^2 + y^2 &> r_0^2
\end{align*}
\]

(12)

where \(x, y\) are the coordinate axes. The initial interface is defined as a round area at the center of the computational domain and standing for the initial nucleus. Undercooled melt enters the domain from the top boundary with a uniform flow \(U\), and exits at the bottom boundary. The initial condition of simulation domain is shown as Fig. 1.

The pure SCN is employed as research object in this paper. The physical properties of SCN and the computational parameters are listed in Table 1.

4. Introduction to SOLA Algorithm, GPU and CUDA

4.1 Brief introduction to SOLA algorithm

Numerical simulation of the filling process is an advanced
4.2 Brief introduction to GPU and CUDA

Driven by the market demand and large scale scientific computing, the programmable Graphic Processing Unit (GPU) has evolved into a highly parallel, multithreaded, manycore processor with tremendous computing power and very high memory bandwidth.\(^{19}\) The floating-point performance and memory bandwidth of the GPU are higher than that of the contemporaneous CPU.

Introduced by NVIDIA in 2006, CUDA is a general purpose parallel computing platform and programming model that regards GPUs as data parallel computing devices to solve many complex computational problems in a more efficiency way than on a CPU.\(^{19}\) CUDA comes with a software environment that allows developers to use C as a high-level programming language. The CUDA programming model assumes a system composed of a host and a device, each with their own separate memory, where the CPU is the host and the GPU is the device. The CPU and GPU work together using the heterogeneous model. Figure 2 shows the work mode of CPU+GPU heterogeneous model.

CUDA threads may access data from multiple memory spaces during their executions. Each thread has private local memory. Each thread block has a shared memory visible to all threads of the block and with the same lifetime as the block. All threads have access to the same global memory. There are also two additional read-only memory spaces accessible by all threads: the constant and texture memory spaces. The global, constant and texture memory spaces are optimized for different memory usages. It is necessary to use them reasonably according to the specific application to improve program performance.

5. The Parallel Implementation of Sola-Phase Field Model on GPU

5.1 Algorithm design and parallel computing process

For the phase-field model solution, the computations of different grid points are data independent at the same time step, and the computations for each grid point are the same. So, the numerical solutions to the whole computational domain meet data parallelism requirement, the computations for eqs. (1), (2), (9) and (10) can be executed on GPU to improve the algorithm’s running speed. Because the data are stored in global memory with the access latency of hundreds to thousands of cycles, and a large number of non-linear memory accesses exist by using finite difference method, the bottleneck of affecting program performance is memory access rather than computation. In this paper, the efficient shared memory (the shared memory latency is almost 1/100 of the global memory latency\(^{20}\)) is employed to achieve coalescent access, and improve memory access speed through copying data from global memory to shared memory only once before parallel computing. In addition, the GPU computations using global memory only and texture memory are also done. When the calculation data are all stored in global memory, any data reading in computation process will access global memory, which spends much time on memory access, resulting in low computational efficiency. When using texture memory in global memory, the original reading data from global memory becomes obtained from texture memory, which has a good acceleration effect on mass data parallelism.\(^{21}\) CUDA GPUs may access data from multiple memory spaces during their executions. Each thread has private local memory.

Fig. 2 The work mode of CPU+GPU heterogeneous model.
Step 3: Calculate the phase-field value at each grid point on GPU.

Step 4: Correct pressure field by using Sola method, and then the correctional pressure value is substituted into eq. (10) to obtain the velocity field.

Step 5: Calculate the temperature field coupled with flow field.

Step 6: Repeat step 3 as the time step increases $\Delta t$ to update microstructure solidification state until the end of the cycle time step.

Step 7: Copy the computing results from device memory to host memory, output the results and make them visible.

5.2 Program Optimization

(1) Computational domain partition. The computational domain can be divided into several small areas according to the CUDA thread hierarchy. A thread block is responsible for the computation of a small area, and each thread in thread block is responsible for a mesh point computing in small area. However, the GPU Stream Multiprocessors (SMs) create, manage, scheduling and execute threads with 32 parallel threads (warp) as a group, as shown in Fig. 4. The data dependences with adjacent threads exist in the computations of the boundary threads in a thread block. Therefore, it is necessary to obtain a ghost cell\(^{22}\) to store data required by boundary threads. For the processed data in a block, the internal data meet coalesced memory access while the data on left and right boundaries need special treatment. Within the permissible range, the larger the blockDim.x is, the smaller the proportion of the data needing special treatment is, which is beneficial to improve program performance. Generally, the number of threads in a thread block is within \([128, 256]\) as a multiple of 32. After testing, the program achieves the optimal performance when the thread block dimension is \((128, 1, 1)\).

(2) Data transmission between the CPU and GPU. In the program, except copying computational data from host memory to device memory before parallel computing, copying computing results from device memory to host memory after completing computation, and being required to output the results at specific time step, GPU is responsible for all the computations of each iteration, in which data exchange with the CPU is not required. In addition, allocating host memory through pinned memory can achieve higher data transmission bandwidth between host memory and device memory.

(3) Global memory access optimization. In order to reduce the use of global memory, the shared memory located in the GPU chip is employed in program to process data. Thus, repeated reading to the overlapping mesh point data from global memory is avoided, and the data reading speed is accelerated. Meanwhile, the non-coalesced access to global memory caused by using difference method is greatly decreased.

6. Simulation Results and Discussions

The simulation platforms are Intel(R) Xeon(R) CPU E5405 @2.00GHz with 4.00GB RAM and a single NVIDIA GeForce GTX 780 GPU using CUDA 5.0 on Windows 7 operating system.

6.1 Dendrite growth evolution process under flow

Figures 5(a)–(b) show the dendritic growth morphology evolution processes when flow velocity $U = 0$ (without thermal noise) and $U = 0.32$ (with thermal noise) for times up to $8600\Delta t$, the time difference between contours is $900\Delta t$ with the grid size $1024 \times 1024$. It can be seen that flow changes dendritic growth symmetry, namely, the length of the upstream main dendrite arm is longer than that of the downstream’s, and the main dendrite arms perpendicular to flow direction grow inclining to the upstream direction. Because the undercooled melt flow takes away the latent heat released during the upstream dendrite solidification, with the downstream latent heat being not easy to spread, which makes the actual undercooling of the upstream dendritic tip greater than that of the downstream’s, and the main dendrite arms perpendicular to flow direction grow inclining to the upstream direction. Because the undercooled melt flow takes away the latent heat released during the upstream dendrite solidification, with the downstream latent heat being not easy to spread, which makes the actual undercooling of the upstream dendritic tip greater than that of the downstream’s, and the main dendrite arms perpendicular to flow direction grow inclining to the upstream direction.
dendrite tip and the upstream dendritic temperature boundary layer thickness vertical to the flow direction is also thinner than the downstream’s. The temperature fields corresponding to Fig. 5(a) when $t = 8600\Delta t$ are shown in Fig. 5(c)–(d).

6.2 Parallel efficiency

In order to study the acceleration effect of the phase-field simulation by utilizing the shared memory, the GPU-based simulations are performed with different memories. Figure 6 shows the computation time at 3000 time steps for different computational domain on a single CPU core and a single GPU by using different memories, where G represents Global Memory; T represents Texture Memory; S represents Shared Memory. The line graph of the speedup compared to the CPU computation is shown in Fig. 7. It can be seen that the computing performance has improved greatly no matter what kind of memory is used. The speedup increases with the grid size increases. The acceleration performance is the highest when using the shared memory, followed by the texture memory, and then the global memory. The GPU computation with the shared memory is 56.16 times faster than that by a single CPU core for $2048^2$ computational grids. This can be interpreted as: the shared memory’s access speed is very fast with only a few clock cycles, and the coalesced access is also realized, which improves the computing performance greatly. The texture memory has cache mechanism, in which the texture cache can pre-fetch several pixels surrounding the coordinate corresponding position. If the probability of hitting the cache is high enough, the computing performance can be better improved. However, when using the global memory, not only is the memory access cycle long, but a large number of memory access operations are also non-coalesced, which affects the computing performance improvement.

6.3 Dendritic tip velocities and tip radius comparisons between CPU and GPU computation

The dendritic tip velocities and tip radius of the upstream direction, the normal to flow direction and the downstream direction change with time are quantitatively compared between CPU and GPU computation under forced flow, as shown in Fig. 8 and Fig. 9. It can be seen that the tip velocities and tip radius based on GPU computation are in good agreement with the simulated values on CPU. In the initial stage of dendrite growth, all directions’ tip velocities and tip radius are larger. With the dendritic growth, the frontier undercooling of dendritic tip decreases rapidly because a large amount of latent heat is released in the solidification process. And then each dendrite tip achieves different stable undercooling, with the tip velocities and tip radius reaching a steady state.

7. Conclusions

(1) A Sola-phase field model combined Sola algorithm with phase-field model is established in this paper. And the GPU-accelerated Sola-phase field simulation is implemented.

(2) The results show that the single GPU computation by utilizing the shared memory reflects the best acceleration performance with a maximum speedup of 56.16, which makes it possible to achieve an efficient phase-field simulation to some extent on a single computer.
The dendritic tip velocities and tip radius calculated on GPU are in good agreement with that on CPU, which demonstrates the reliability of the parallel computing method for Sola-phase field simulation based on GPU.

Acknowledgments

This Project is Supported by the National Natural Science Foundation of China (51161011, 11364024), the Key Technology R&D Program of Gansu province (1204GKA065), the Fundamental Research Funds for the Universities of Gansu province (201210) and the Funds for Distinguished Young Scientists of Lanzhou University of Technology (J201304).

REFERENCES