Gazizova Daria
Shumilov Alexander
Zorina Veronika
20 Dec. 2019

Optimization of solving PDE using explicit finite differences method

Project on the course Introduction to Linux and Supercomputers

Abstract

This work devoted to solving partial differential equation describing changing of order parameter of the solidification process using explicit finite differences method. Also to optimize the program OpenMP for parallalization and different optimization flags during compilation were used. Code for this problem was written in C and running on Zhores sandbox. Vizualization of the process was done in Paraview.

Maintenance

Introduction

Partial Differential Equation

The Ginzburg-Landau kinetic method, which takes into account the time dependence of the order parameter, is used to simulate the structural phase transformations of solidification recrystallization.

The free energy functional in the Ginzburg-Landau kinetic method in the simplest case of a one-component system has the form:

$$G[\varphi] = \int d^3\boldsymbol{r}\left(f(\varphi, T) + \frac{1}{2}\left|\sigma \nabla \varphi\right|^2\right),$$

where $\varphi(\boldsymbol{r}, t)$ - an order parameter, which depends on time and position, σ - a surface energy, $f(\varphi, t)$ - a function of the order and temperature parameter having local minima at values φ corresponding to the liquid and solid phases.

Taking the variation of free energy by order parameter we have the following formula

$$\frac{\partial \varphi}{\partial t} = -K\left(\frac{\partial f}{\partial \varphi} - \sigma^2 \Delta \varphi\right),$$

where K is parameter determining phase separation mobility and $\frac{\partial f}{\partial \varphi}$ in the simplest case looks like

$$\frac{\partial f}{\partial \varphi}(\varphi, T_m) = h\left(\left(\varphi - \frac{1}{2}\right)^3 - \frac{1}{4}\left(\varphi - \frac{1}{2}\right)\right),$$

h - is a system parameter.

Explicit method of finite differences

In our case Partial Differential Equation in $Q_T = [0,X] \times [0,T]$ looks like:

$$\frac{\partial \varphi}{\partial t} = -K(f(\varphi) - \sigma^2 \Delta \varphi), x \in R^3, f(\varphi) = (\varphi - \frac{1}{2})^3 - \frac{1}{4}(\varphi - \frac{1}{2}),$$

with the next conditions

$u(x, o) = u_0(x), u(0,t) = \mu_1(t), u(X, t) = \mu_2(t)$

Let's $\sigma = 1$ and K = 1, then implementation of Finite elements method is:

$$\frac{u_{ijk}^{n+1} - u_{ijk}^n}{\tau} = \frac{u_{(i-1)jk}^n - 2u_{ijk}^n + u_{(i+1)jk}^n}{h^2} + \ldots - f(u_{ijk}^n)$$

$$u_{ijk}^{n+1} = u_{ijk}^n + \tau\frac{u_{(i-1)jk}^n - 2u_{ijk}^n + u_{(i+1)jk}^n}{h^2} + \ldots - \tau f(u_{ijk}^n)$$

Implementation

The realization of the method describing above is implemented in equation.c file. on Zhores supercomputer. Then all files were transfered (using scp command) and vizualized using Paraview. Video order_parameter_0_5_surface.avi was constructed to show evolution of the process. As an output there are *.vtk format files for each timestep (timesteps = 10).

For computing times file equation_cluster.c was used. Launching of this project is the same so as compilation and input arguments but output is running time. File equation_cluster.c use as input different number of threads (1 4 6 8 12) and with two optimization flags (-g, -Ofast) while complilation. It takes number of threads and matrix size as an input. During implementation 64, 128, 512 size matrix were used. On Zhores we used parallel.sh to run executable file.

Example of compiling and running files on Zhores.

```
gcc equation_cluster.c  -lm -fopenmp -o parallel_g.exe
sbatch -N 1 -p cpu -c 24 parallel.sh
```

Results

Running times for all optimization flags and sizes of computing matrices are in file parallel4.txt. As it was shown, the result of optimization is not so significant with -g flag, but it definitely counts while using flag -Ofast especially for large number of threads and with matrix with large size. From file with output running time we could see that after compilation with -g flag there is no acceleration even for many threads, but after compilation with -Ofast flag acceleration was achieved. It could be because -Ofast flag provide optimization more for code size and execution time with fast none accurate math calculations.

Attachments

equation.c

equation_cluster.c

parallel.sh

order_parameter_0_5_surface.avi

parallel4.txt

parallel3_g.exe

parallel3_Ofast.exe