PARALLEL PARTICLE MOVEMENT SIMULATION ALGORITHM BASED ON HETEROGENEOUS COMPUTING*

L. G. Zhang[†], K. J. Fan, J Huang, J Yang, K. F. Liu, W. Qi, L. Cao
Institute of Applied Electromagnetic Engineering, Huazhong University of Science and Technology, Wuhan, China

Abstract

Particle in cell (PIC) algorithm studies the self-consistent motion of multi-particle system by solving equations of particle dynamics, this algorithm is widely used to evaluate the nonlinear space charge effect of the high intensity or low energy beam. In order to reduce the random noise in the simulation, a huge number of particles should be traced, the process expends many computer hardware resources and a lot of computing time. Heterogeneous computing can greatly improve the efficiency of large quantities of the particle tracking by making full use of different types of computing resources. In this paper we give the algorithm which uses both CPU and GPU to trace the particles in electromagnetic field. The results show that the given algorithm increases the efficiency significantly.

INTRODUCTION

PIC algorithm evaluates the motion of the multi-particle system through solving the dynamic equation of particles. After that, the motion of the system is given in the statistical results. The PIC algorithm approaches the accurate result when the particle number is large and the space grid is small [1]. In order to reduce the statistical error of the result, using a huge number of particles in the PIC algorithm is necessary. Some studies [2] show that it is enough to obtain a good result by using one "super particle" to describe 10^4-10^9 particles.

PIC algorithm consists of four steps in a time step. Firstly, the electromagnetic field is evaluated, then the calculated value is interpolated at the positions of the particles. Using this values, the motion of the particles are integrated. Finally, the statistical results on the grid is given. The processing of the PIC algorithm in one time step is shown in figure 1. The name of the particle is given by index *i*, and the field on each spatial grid is labeled with index *j*. The particle dynamic equation describe the position and velocity taking on all values in ***x*** and ***v*** space which called phase space.

* Work supported by in part by the 2011 Project- Hubei Collaboration Innovation Center of Non-power Nuclear Technology and in part by National Program for Study of the Key Techniques in High-voltage Power Supply Using Ferrite Insulated Core Transformer under grant 11305068 through the ministry of science and technology, China.

† Email address: zhanglie@hust.edu.cn

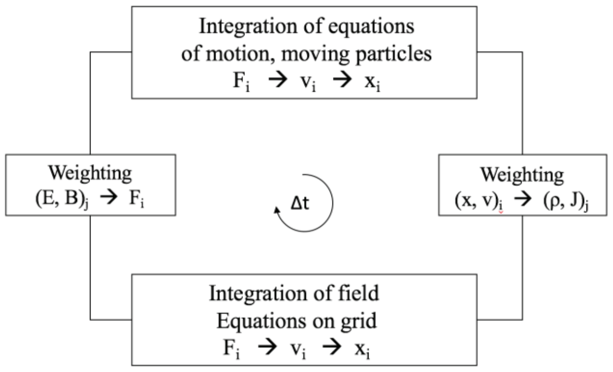
Integration of equations
of motion, moving particles
$F_i \rightarrow v_i \rightarrow x_i$
Weighting
$(E, B)_j \rightarrow F_i$
Δt
Weighting
$(x, v)_i \rightarrow (\rho, J)_j$
Integration of field
Equations on grid
$F_i \rightarrow v_i \rightarrow x_i$

Figure 1: PIC algorithm in one time step

In one time step, the moving particle calculations of each particle are the same. Firstly, interpolate the electric field intensity and magnetic flux density at each particle position, then the moving information of each particle in next time step are obtained by integrating the equation of particle motion. Interpolation algorithm in the GPU is operated by computing hardware which can finish the interpolation algorithm in a single GPU instruction cycle. Thus, the operation speed is much higher than that run by software. While the multi core structure of GPU is very suitable to integrate the multi particle motion in parallel. In this paper, based on the OpenCL programming framework, CPU is used to control the data storage and search tasks, then it controls GPU computing hardware unit to interpolation electric field intensity and magnetic flux density, finally integrate the particles dynamic equation on the GPU compute kernels in parallel. This calculation framework can greatly improve the efficiency of the particle movement simulation algorithm.

ALGORITHM OF MOVING PARTICLES

When we integrate the equations of particle dynamic, we should obtain the electric field intensity and magnetic flux density at the particle position. After that, the equations of particle dynamic as shown in Eq.(1) should be integrated. Many method could be used to integrate the equations of particle dynamic. This part of calculation should execute with high efficiency and acceptable accuracy.

$$\begin{cases} m\dfrac{d^2\mathbf{r}}{dt^2} = q\mathbf{E} + q\mathbf{v}\times\mathbf{B} \\ \mathbf{v} = \dfrac{d\mathbf{r}}{dt} \end{cases} \tag{1}$$

Leap-frog Method

The leap-frog method is commonly used in the PIC code, since its efficiency is the highest. This method integrate two first-order equation separately for the particles, as shown below,

$$\begin{cases} m\mathbf{v}' = \mathbf{F} \\ \mathbf{x}' = \mathbf{v} \end{cases} \tag{2}$$

Where **F** is the force which can be calculate as

$$\mathbf{F} = \mathbf{F}_{electric} + \mathbf{F}_{magnetic} = \mathbf{E} + q\mathbf{v}\times\mathbf{B} \tag{3}$$

The equations can be replaced by the finite-difference equations below,

$$\begin{cases} m\dfrac{\mathbf{v}_{new} - \mathbf{v}_{old}}{\Lambda t} = \mathbf{F}_{old} \\ \dfrac{\mathbf{x}_{new} - \mathbf{x}_{old}}{\Lambda t} = \mathbf{v}_{new} \end{cases} \tag{4}$$

The leap-frog method only keeps the position and velocity information for each particles in the one previous time step. Thus this method also request the lowest storage capability of the computer. In the leap-frog method, position and velocity only contains the first-order accuracy, so the time step for each calculation cycle should be small to ensure the accuracy of the integration.

High Order Method

If we need more accuracy in the integration of the particle dynamic equation, high order methods are necessary. Using a high-order method would also multiply the operations taken for each particle. Runge-Kutta method is a high-order method in which the accuracy increases with the order. In some PIC code, fourth order Runge-Kutta method is used. For a problem in Eq.(5), Eq.(6) shows the calculation method of the fourth order Runge-Kutta method.

$$\begin{cases} y' = f(x,y) \\ y(a) = y_0 \end{cases}$$

(5)

$$\begin{cases} y_i = y_{i-1} + \dfrac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 = f(x_{i-1}, y_{i-1}) \\ K_2 = f(x_{i-1} + \dfrac{h}{2}, y_{i-1} + \dfrac{h}{2}K_1) \\ K_3 = f(x_{i-1} + \dfrac{h}{2}, y_{i-1} + \dfrac{h}{2}K_2) \\ K_4 = f(x_{i-1} + h, y_{i-1} + hK_3) \end{cases}$$

(6)

Runge-Kutta is a method using Taylor expansion indirectly which prevent the calculation of the derivation of *f*(x). But the electric field intensity and the magnetic flux density should interpolated four times in the calculation. And the velocity should be integrated four times. Hence, the fourth order Runge-Kutta method expands more computer hardware resources.

Linear Multistep Method

The leapfrog and the Runge-Kutta methods are single step method, in which the particle velocity and the position is only calculated from the previous steps. The linear multistep method is a method which combine the velocities and positions in several previous time steps. The displacement ***r*** can be calculated using [3]

$$\boldsymbol{r}_{n+1} = \alpha_0\boldsymbol{r}_{n-1} + \alpha_1\boldsymbol{r}_n + \Delta t(\beta_0\boldsymbol{r}'_{n-1} + \beta_1\boldsymbol{r}'_n) + \Delta t^2(\gamma_0\boldsymbol{r}''_{n-1} + \gamma_1\boldsymbol{r}''_n) \tag{7}$$

Where α_0, α_1, β_0, β_1, γ_0, γ_1 are the coefficients to be determined. Expand Eq.(7) in Taylor series, the displacement $\boldsymbol{r}_{n+1}$ can be described only by $\boldsymbol{r}_n$. The displacement in the previous time step $\boldsymbol{r}_{n-1}$ can also be described only by $\boldsymbol{r}_n$ in form of Taylor series.

$$\begin{aligned} \bar{\boldsymbol{r}}_{n+1} &= \boldsymbol{r}(t_n + \Delta t) \\ &= \boldsymbol{r}_n + \boldsymbol{r}_n\Delta t + \frac{\boldsymbol{r}''_n}{2!}\Delta t^2 + \frac{\boldsymbol{r}^{(3)}_n}{3!}\Delta t^3 + \frac{\boldsymbol{r}^{(4)}_n}{4!}\Delta t^4 + \frac{\boldsymbol{r}^{(5)}_n}{5!}\Delta t^5 + o(\Delta t^5) \\ &= (\alpha_0 + \alpha_1)\boldsymbol{r}_n + (-\alpha_0 + \beta_0 + \beta_1)\boldsymbol{r}'_n\Delta t \\ &+ (\frac{\alpha_0}{2} - \beta_0 + \gamma_0 + \gamma_1)\boldsymbol{r}''_n\Delta t^2 \\ &+ (-\frac{\alpha_0}{6} + \frac{\beta_0}{2} - \gamma_0)\boldsymbol{r}^{(3)}_n\Delta t^3 \\ &+ (\frac{\alpha_0}{24} - \frac{\beta_0}{6} + \frac{\gamma_0}{2})\boldsymbol{r}^{(4)}_n\Delta t^4 \\ &+ (-\frac{\alpha_0}{120} + \frac{\beta_0}{24} - \frac{\gamma_0}{6})\boldsymbol{r}^{(5)}_n\Delta t^5 + o(\Delta t^5) \end{aligned} \tag{8}$$

Thus, we can get the undetermined coefficients α_0, α_1, β_0, β_1, γ_0, γ_1 by Eq. (8).

As for the velocity, it is the first order derivation of the displacement, it can also be calculated using the Taylor expansion method.

The precision of the linear multistep method is much better than the leapfrog method and more efficient than fourth order Runge-Kutta method.

PROGRAM STRUCTURE BASED ON HETEROGENEOUS COMPUTING

Heterogeneous computing handles the calculation task using more than one kind of processors or cores. These systems gain performance by many special processors [4]. To obtain the electric field intensity and magnetic flux density at the particle location, the GPU is used to interpolate the values in 3 dimension space. Because of the hardware calculator in GPU for interpolation, this job can be completed in one GPU instruction cycle which makes it much faster than software interpolation. While multi-core structure of the GPU makes it very suitable to handle multi-particle motion calculations in parallel. High performance of the CPU makes it suitable to handle complex tasks, such as controlling the work flow and managing the computing resources. In this paper, CPU is used to control the data storage and search, then call GPU computing hardware unit to interpolation electric field intensity and magnetic flux density, finally integrate the particles dynamic equation in parallel. This calculation framework can greatly improve the program efficiency. The program flow chart is shown in figure 2.

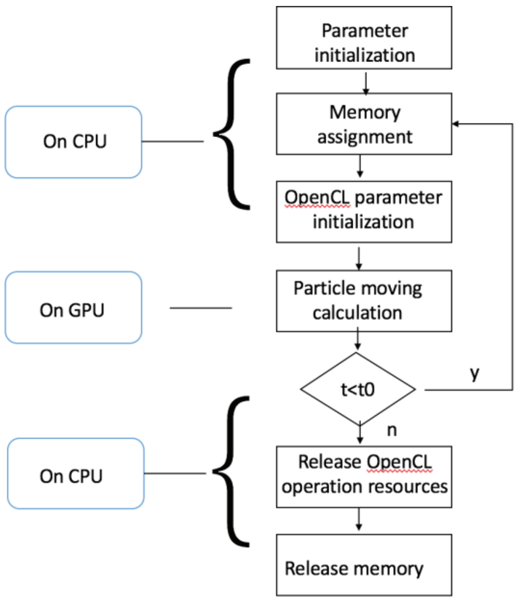
Parameter initialization
Memory assignment
OpenCL parameter initialization
Particle moving calculation
t<t0
y
n
Release OpenCL operation resources
Release memory
On CPU
On GPU
On CPU

Figure 2: Program flow chart

Open Computing Language (OpenCL) is a framework for writing programs that execute across heterogeneous platforms. The OpenCL execution model supports data parallel and task parallel programming models, as well as supporting hybrids of these two models [5]. In this paper, global memory is used to store the field values and particle motion values. CPU control the memory and the data flow, and many computing kernels execute the interpolation and particle motion integrating tasks on GPU. The compute device here is GPU and the host device here is CPU.

RESULTS AND DISCUSSION

In this paper, we have run the programs on the platform of MacBook Pro with the operation system of OS X Yosemite. The OpenCL version is 1.2. The hardware information of the computer is shown below in table1.

Table 1: Hardware Information of the Computer

Device	Type	Parameters
CPU	Intel Core i5-4258U	4 Core @ 2.4 GHz
GPU	Intel Iris Pro Graphics 5100	832 GFLOPS
RAM	DDR3	8 GB @ 1600 MHz

The program has been run on single CPU core using serial algorithm, 4 CPU cores using parallel algorithm, and GPU using heterogeneous computing algorithm. Speed of the algorithms are shown in figure 3. It shows that the parallel algorithm cost more time than serial algorithm when the particle number is small as a result of the long time used to assign the memory and resources. This part of time is more than that used to integrate the particle movement. With the particle number increasing, the integration cost more and more time which would dominate the total time, the advantage of the parallel algorithm efficiency become more and more obvious. When the particle number reaches the level of 10^7, speed of the heterogeneous computing algorithm running on the GPU is 20 times faster than serial algorithm.

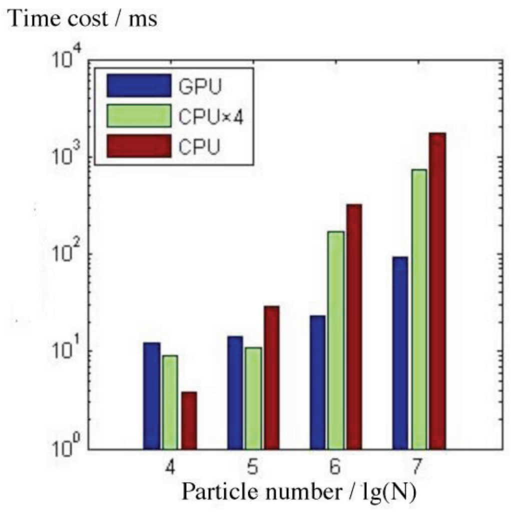

Figure 3: Results of the three methods

CONCLUSION

Linear multistep method is used in particle moving integration, the precision of this method is much better than the leapfrog method and more efficient than fourth order Runge-Kutta method. Then this method is realized by heterogeneous computing program, in which CPU is used to control the data storage and search tasks, and GPU computing hardware unit to interpolate electric field intensity and magnetic flux density and integrate the particles dynamic equation in parallel. The efficiency of the particle movement simulation significantly increased with these methods.

ACKNOWLEDGEMENT

This work was supported the 2011 Project- Hubei Collaboration Innovation Center of Non-power Nuclear Technology and in part by National Program for Study of the Key Techniques in High-voltage Power Supply Using Ferrite Insulated Core Transformer under grant 11305068 through the ministry of science and technology, China.

REFERENCES

[1] C. K. Birdsall and A. B. Langdon, *Plasma Physics via Computer Simulation*, NY, USA: IOP Press, 1991.

[2] T. Arzt, “Numerical simulation of the RF ion source RIG-10,” *J. Phys. D, Appl. Phys.,* vol. 21, no. 2, p. 278, 1988.

[3] H. W. J. Lenferink, “Contractivity preserving explicit linear multistep methods,” *Numer. Math.*, vol. 55, no. 2, pp. 213–223, 1989.

[4] A. Shan, "Heterogeneous Processing: a Strategy for Augmenting Moore's Law." *Linux Journal,* vol.142, pp. 80-82, 2006.

[5] Khronos Group Inc, `https://www.khronos.org/registry/cl/sdk/1.2/docs/man/xhtml/`