PARALLEL BILINEAR SPATIAL INTERPOLATION ALGORITHM BASED ON GPGPU

[1]WU QINGSHUANG, [2]WANG QIANG, [3]CHENG XIANGFU

[1] College of Territorial Resources and Tourism, Anhui Normal University, Wuhu Anhui, 241000, China
[2]City Construction Academy, Hunan City University, Yiyang Hunan, 413000, China
[3]Anhui Key Laboratory of Natural Disaster Process and Prevention, Wuhu Anhui, 241000, China
E-mail: qepwq_wu@sina.com

ABSTRACT

In view of the problem that massive spatial data interpolation is a complex and time-consuming computing process, and the traditional CPU implementation methods can't meet the real-time processing demand, in this paper, we propose a parallel bilinear spatial interpolation algorithm, which is accelerated by the graphic processing unit(GPU) and implemented in compute unified device architecture(CUDA). Firstly, we introduce the basic idea of general purpose computing on graphics processing units (GPGPU) and then discuss the technology of the CUDA programming model. Secondly, we introduce the principle of the bilinear interpolation algorithm and analyze the feasibility of mapping the bilinear interpolation algorithm program onto the GPU, and then we provide detail of implementing our parallel bilinear spatial interpolation algorithm on GPU that uses the CUDA programming model. Finally, we conduct several groups of experiments to demonstrate the strength of our GPU implementation method by measuring the performance over standard CPU implementation. The experimental results show that the GPGPU-based parallel algorithm can take full advantage of the GPU's parallel computing capabilities, and can achieve about 40 times speedup; it is able to meet the demand of real-time processing of massive spatial data interpolation.

Keywords: *General Purpose Computing on Graphics Processing Units; Compute Unified Device Architecture; Bilinear Spatial Interpolation; Spatial Index; Speedup*

1. INTRODUCTION

Spatial data interpolation is an important operation in Geographic Information System(GIS), which can calculate the values of unspecified location point from the values of the known by a certain interpolation algorithm[1]. With GIS widely using in various areas, spatial data's distribution and density have become increasingly demanding. Spatial data interpolation is causing people more and more attention[2].

At present, many scholars have launched a large number of studies on spatial interpolation, and the major research interests were focused on spatial interpolation methods[3-5] and spatial interpolation applications. Various types of spatial interpolation methods such as geometric method, statistical method, function method, physical model simulation method, stochastic simulation method, spatial statistical method and integrated method were proposed and applied to the DEM terrain spatial interpolation[6], air pollution measurements[7], soil spatial interpolation[8], rainfall spatial interpolation [9] and many other spatial analysis applications. In these existing studies, the researcher is more concerned about the accuracy of the spatial interpolation results. In practical applications, the spatial interpolation task is often a compute-intensive processing and needs powerful computing capabilities. With spatial interpolation more and more applying to model-complex and data-massive spatial analysis which usually requires a lot of computing time, the time efficiency of the spatial interpolation algorithms is causing people more and more attention. In this case, to study how to use the new parallel computing equipment to speed up the spatial interpolation processing has a very important significance.

In the past, mainly means of traditional parallel processing in GIS included multi-core CPU computing, computer cluster computing or grid computing, which only use the CPU as the computing components. With the slowdown of the upgrade of the CPU's clock frequency, the cost of parallel processing performance improvement becomes more and more big. Recently, the performance and capabilities of the graphics processing unit (GPU) have been a marked increase. Today, the modern GPU is not only a powerful graphics engine but also a highly parallel programmable processor featuring peak arithmetic and memory bandwidth that substantially outpace its CPU counterpart[10]. The GPU's rapid increase in both programmability and capability has spawned a research community that has successfully mapped a broad range of computationally demanding, complex problems to the GPU. This class of research that development of no graphical applications on graphics accelerators is referred to as general-purpose programming on GPUs (GPGPU)[11]. In present, the main GPU manufacturers NVIDIA and ATI have launched their own GPGPU solution each other. These solutions can run the users' code which is designed for the problem domain on the GPU directly. In contrast, NVIDIA's CUDA (Compute Unified Device Architecture) solution is more mature.

In this paper, in order to take use of the overwhelming computing power of the CPU to accelerate the spatial interpolation, we propose and implement a CUDA-based parallel bilinear interpolation algorithm. The experimental results show that the CUDA-based parallel algorithm can achieve about 40 times speedup compared with the traditional CPU implementation method. The paper is organized in 6 sections. In Section 2, we discuss the basic idea of the general purpose computing on graphics processing units and the technology of the CUDA programming model. In Section 3, we introduce the principle of the bilinear interpolation algorithm and analyze the feasibility of mapping the bilinear interpolation algorithm program onto the GPU, and then we provide detail of implementing our parallel bilinear spatial interpolation algorithm based on GPGPU. In Section 4, we demonstrate the strength of our GPU implementation method by measuring the performance over standard CPU implementation. Finally, we summarize the implementation issues on the GPU and give thanks.

2. GPGPU WITH CUDA

Traditional GPGPU programming is done directly through graphics APIs(OpenGL or DirectX). Although many researchers are successful in getting applications to work through these graphics APIs, there is a fundamental mismatch between the general programming method and the graphics APIs programming method. In this graphics APIs programming method, the programmers are not only required to be familiar with the parallel algorithms, but also need to understand the graphics hardware programming interface which has greatly restricted the development of GPU computing.

With the continuous improvement of GPGPU, in order to make the GPGPU programming better, the main GPU manufacturers and technology alliances have launched their own GPGPU solutions. In 2006, NVIDIA launched the CUDA programming model which greatly improves the programmability and versatility of GPGPU programming. The CUDA is a co-evolved hardware-software architecture that can provide a C-like programming environment and programming language for the programmer. The basic idea of CUDA programming model is mapping the general-purpose computation onto the GPU uses the graphics hardware in much the same way as any standard graphics application, and as much as possible to develop thread-level parallelism[12].

As it is showed in figure 1, a CUDA program can be divided into two parts: the host-side code and the device-side code. Typically, the host-side code executes on the CPU serially, and device-side code runs parallel on the GPU. Here, CPU and GPU constitute a heterogeneous computing system: the CPU can be deemed as a control termination which can perform the data organizing, task scheduling and the serial code of the program. Then, the GPU can be deemed as a super-large-scale data parallel co-processor which is responsible for the part of the CUDA program that can be parallel executed.

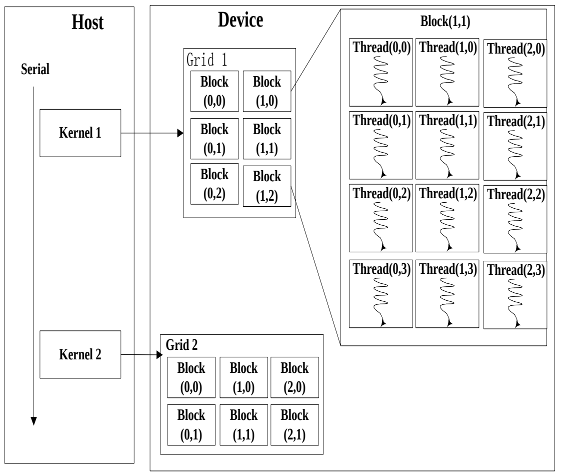


Figure 1: CUDA Programming Model

3. GPU-ACCELERATED PARELLEL BILINEAR SPATIAL INTERPOLATION ALGORITHM

3.1 Bilinear Spatial Interpolation Algorithm

In bilinear interpolation algorithm, the value of the interpolation point can be calculated by a bilinear polynomial function which is determined by 4 sample points nearby the interpolation point. The bilinear polynomial function can be expressed as:

$$f(x,y) = a_0 + a_1 x + a_2 y + +a_3 xy \quad (1)$$

It can be solved for the four coefficients, through substituting the values of the four sample points data which are closest to the interpolation point into formula 1, and substituting the plane coordinates of the interpolation point, then we can calculate out the interpolation point's elevation value.

Bilinear interpolation can be approximated regarded as the extension of the linear interpolation function with two variables, which carry out a linear interpolation in two directions of x, y. As it is shown in figure 2, assuming the function values of $f(x_1,y_1)$、$f(x_1,y_2)$、$f(x_2,y_1)$ and $f(x_2,y_2)$ are known, then to calculate the value of $f(x,y)$ at the point $P(x,y)$.

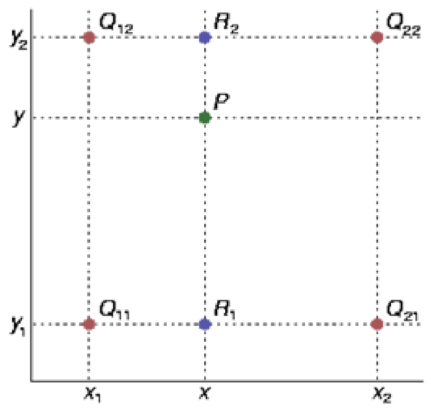


Figure 2: The Schematic Diagram of Bilinear Interpolation Algorithm

First, performing a linear interpolation in the X direction, we can get the following formulas:

(2)

(3)

Then, performing a linear interpolation in the Y direction, we can get the following formula:

.

(4)

Substituting the formula 2,3 into the formula 4, then the result of $f(x,y)$ can be obtained.

(5)

The final results of bilinear interpolation have nothing to do with the interpolating order, changing the order of x, y interpolating, the results will not change. The bilinear interpolation algorithm can get a high quality and good continuity interpolating result. It is particularly suitable for grid-based digital terrain model interpolation and it is widely used in GIS.

3.2 Parallelism Analysis of Bilinear Spatial Interpolation Algorithm

According to the description of the bilinear interpolation algorithm, we can know that the value of each interpolation point is only related to its nearest neighbor four sampling points, and the whole processing has a good data independence and task parallelism. In practical application, there are often a large number of interpolation points need to be calculated. For such computing-intensive and task-parallel interpolation processing, we can make use of the GPU overwhelming computing power to accelerate the processing.

3.3 Design and Implementation of Bilinear Spatial Interpolation Algorithm Based on GPGPU

3.3.1 Sampling data organization and spatial index

In the processing of bilinear interpolation, it is necessary to search for the four sampling points which are most nearest to the interpolation point. If the sampling data set is discrete and having no distribution-law, it must be searched in the entire sampling point sets. When the amount of the sampling points is quite large, such searching progress is very inefficient. To improve the efficiency of lookup, it needs a good sampling data organization and spatial index.

Searching for the nearest k points in the vicinity of a specific point is called k-nearest neighbor problem. In this paper, we use the space partition strategy to search for the nearest k points, that is to say, divide the sampling data set into the data block and then create a grid index for the each data block, so as to reduce the scope of the data search to speed up the search speed.

The main process of creating spatial data block index is as follows: (1) Calculate the number of sampling points and the area of the sampling region, then get the density of the sampling points.(2) According to the data density and the average number of sampling points in each sub-block, determining the number of rows and columns of the grid sub-blocks. (3) Traversing all the sampling points of the interpolation region, and partitioning them into the respective data sub-block.

As shown in figure 3, after the creating of spatial grid index, the process of searching for the k-nearest neighbor point of a specific interpolation point is as follow: First, determine the sub-block of the interpolation point, and then search for the k-nearest neighbor point in the sub-block and its adjacent sub-blocks.

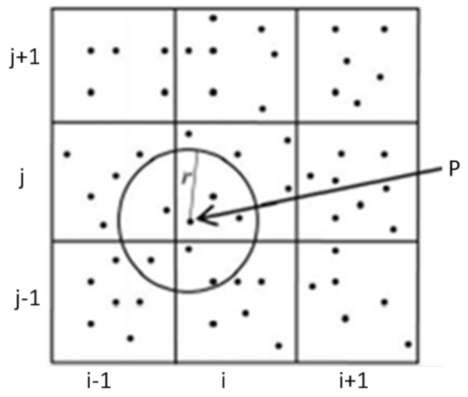


Figure 3: Space Partitioning and Searching Process

3.3.2 Design and implementation of bilinear spatial interpolation algorithm based on GPGPU

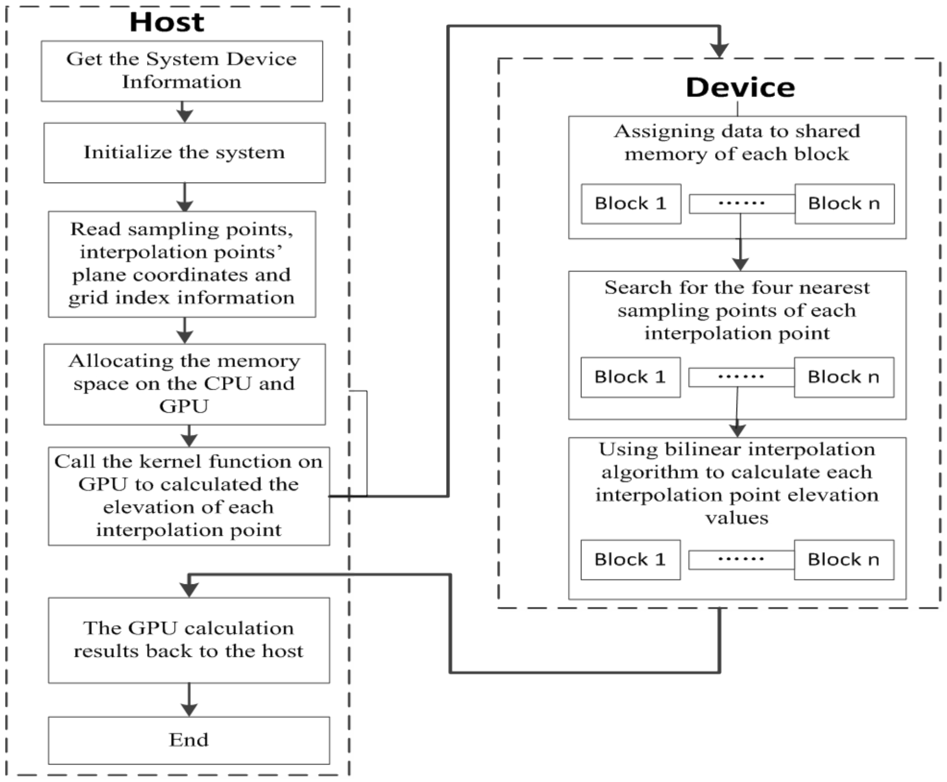

Host
Get the System Device Information
Initialize the system
Read sampling points, interpolation points' plane coordinates and grid index information
Allocating the memory space on the CPU and GPU
Call the kernel function on GPU to calculated the elevation of each interpolation point
The GPU calculation results back to the host
End
Device
Assigning data to shared memory of each block
Block 1
......
Block n
Search for the four nearest sampling points of each interpolation point
Block 1
......
Block n
Using bilinear interpolation algorithm to calculate each interpolation point elevation values
Block 1
......
Block n

Figure 4: The Flow Chart of Bilinear Spatial Interpolation Algorithm Based on GPGPU

As shown in figure 3, the main process of the GPU-accelerated bilinear spatial interpolation algorithm can be described as follows:

(1) Read the data of the sampling points, interpolation points' plane coordinates and grid index into the host memory, and apply for the memory space with the size of the data in the GPU's global memory, then transmit the data from host to the GPU.

(2) Set the kernel functions configuration parameters that are to determine the dimension of the grid and block. Due to the limitation of the size of the GPU's shared memory which is only 32KB, the parameters should be reasonably considered according to the template size, search image size and number of the search.

(3) Call the kernel function on GPU to calculate the elevation of each interpolation point: □ ①Transfer the data from host memory to the device global memory by cudaMemecpy( ) function, then assigning the relevant data to the shared memory of each block. ②Parallel searching for the four nearest sampling points of each interpolation point. ③Parallel using the bilinear interpolation algorithm to calculate the values of each interpolation point's elevation.

(4) Write the results obtained by the GPU's processing to the global memory and then return to the host memory.

4. EXPERIMENT AND DISCUSSION

4.1 Experimental Environment

The experimental computing platform configuration and test environment are as follows:

(1)Hardware environment: The CPU is Intel core i3 550; and the GPU is NVidia GeForce 410M.

(2) Software environment: The operating system is Microsoft Windows 7; and the program development tools are Visual Studio 2010 and CUDA SDK 4.0.

4.2 Experimental Results and Analyzing

Using a somewhere 3D point cloud sets as the experimental data, as it is shown in figure 5, the point cloud sets is composed by a series of homogeneous distribution three-dimensional points.

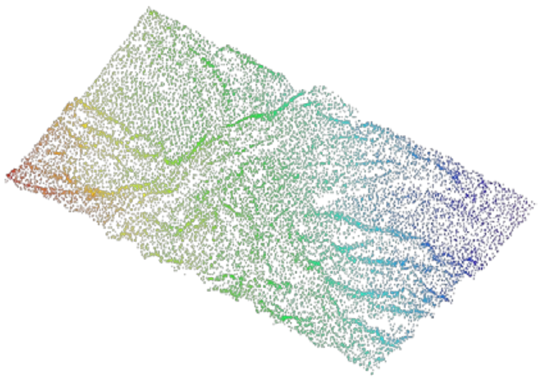


Figure 5: The Three-Dimensional Sampling Point Sets Of Somewhere Terrain Data

Perform spatial interpolation to generate the grid DEM respectively by using the CPU serial algorithm and the GPU parallel algorithm. Taking into account the different data amount experiment need, we performed 7 comparative experiments, the size of the generated grid DEM were 40×40、100×100、200×200、300×300、500×500、800×800、1200×1200. In order to compare their performance, we recorded the time-consuming of the two algorithms which is the mean of 10 times respectively. Table 1 shows the time-consuming of serial and parallel algorithms of different grid size.

Table 1: The Time-Consuming Of Serial And Parallel Bilinear Spatial Interpolation Of Different Grid Size

No.	grid size	CPU serial algorithm (s)	GPU parallel algorithm (s)
1	40×40	1.70	0.11
2	100×100	10.34	0.37
3	200×200	41.43	1.21
4	300×300	93.13	2.36
5	500×500	240.02	5.70
6	800×800	630.92	14.40
7	1200×1200	1432.33	32.41

As showed in figure 6, it can be seen that the speed of the GPU-accelerated parallel algorithm is far more than 10 times faster than the conventional CPU algorithm. When the generated grid DEM size is small, the speedup is relatively low because the GPU's parallelism computing power can't be took full advantage of, as there is communication cost between device memory and host memory. With the generated grid DEM size increasing, the speedup is continuously improving; and it can reach about 40 times eventually. It indicates that GPGPU technology is especially suitable for these high parallelism and computation-intensive applications.

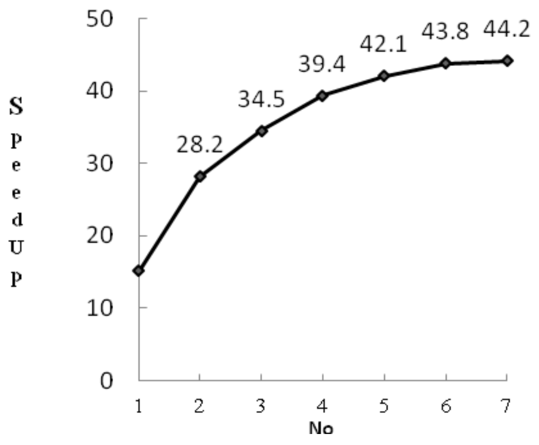
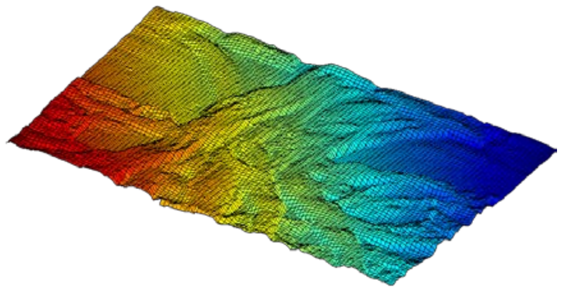
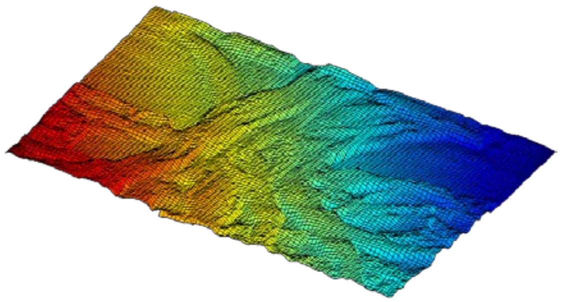


Figure 6: The Speedup Of Different Resolution Image

Figure 7 shows the results of 100×100 generated grid DEM respectively by using the CPU serial algorithm and the GPU parallel algorithm. From contrast of the grid DEM 3D visualizations, it is can be seen that there is almost no difference which human eye can observe between the CPU interpolation results and the GPU interpolation result. The generated grid DEM is continuous, strong fidelity.



(A) The Result Of CPU Interpolation



(B) The Result Of CPU Interpolation

Figure 7: The Results Of 100×100 Generated Grid DEM By Bilinear Spatial Interpolation Algorithm

Figure 8 shows the relative error of each generated grid point's elevation between the CPU interpolation results and the GPU interpolation result. It is can be seen that the relative error is very small. The result shows that the precision using GPU computing can meet the demand of spatial interpolation, the parallel algorithm is correct and achieve the desired effect.

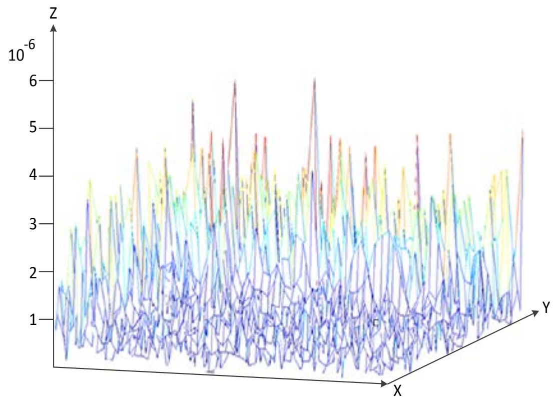


Figure 8: The Relative Error Of Each Generated Grid Point's Elevation Between The CPU Interpolation And The GPU Interpolation.

5. CONCLUSIONS

The modern GPU is not only a powerful graphics engine but also a highly parallel programmable processor, which can provide hundreds of computing cores to run thousands of threads. With the development of GPGPU, how to use the GPU's powerful parallel computing performance to speed up the computing-intensive and time-consuming task has become a research hot in GIS. NVIDIA's CUDA programming model provides a powerful programming environment and instruction set, which greatly reduce the difficulty of GPU programming and accelerates the popularity of GPGPU. In this paper, in order to take use of the powerful computing power of the CPU to accelerate the spatial interpolation, we propose a CUDA-based parallel bilinear spatial interpolation algorithm and provide the algorithm implementing detail. The experimental results show that the GPGPU-based parallel algorithm can take full advantage of the GPU's parallel computing capabilities, and can achieve about 40 times speedup; it is able to meet the demand of real-time processing of massive spatial data interpolation.

6. ACKNOWLEDGMENT

This work was supported by Hunan Science and Technology Program(No. 2011SK3130), Yiyang

Municipal Science and Technology Program(No.YK0956), Anhui provincial physical geography and human geography key disciplines foundation (No. Asdg1103), National Natural Science Foundation of China(No. 41271516).

REFRENCES:

[1] Miller H.J, "Tobler's first law and spatial analysis", Annals of the Association of American Geographers, Vol. 94, No. 2, 2004, pp. 284-289.

[2] Lam, Nina Siu-Ngan, "Spatial Interpolation Methods: A Review", Cartography and Geographic Information Science, Vol.10, No 2, 1983, pp.129-150.

[3] George Y. Lu1, David W. Wong, "An adaptive inverse-distance weighting spatial interpolation technique", Computers & Geosciences, Vol. 34, No.9, 2008, pp.1044–1055.

[4] Jonathan A. Greenberg, Carlos Rueda, Erin L. Hestir, etc, "Least cost distance analysis for spatial interpolation", Computers & Geosciences, Vol. 37, No. 2, 2011, pp. 272–276.

[5] Hannes Kazianka, Jürgen Pilz, "Bayesian spatial modeling and interpolation using copulas", Computers & Geosciences, Vol. 37, No. 3, 2011, pp.310-319.

[6] Christopher W. Bater, Nicholas C. Coops, "Evaluating error associated with lidar-derived DEM interpolation", Computers & Geosciences Vol. 35, No. 2, 2009, pp.289–300.

[7] Stijn Janssena, Gerwin Dumontb, Frans Fierensb, etc, "Spatial interpolation of air pollution measurements using CORINE land cover data", Atmospheric Environment, Vol. 42, No. 20, 2008, pp. 4884–4903.

[8] Harley T. Davisa, C. Marjorie Aeliona, d, Suzanne McDermottb, etc, "Identifying natural and anthropogenic sources of metals in urban and rural soils using GIS-based data, PCA, and spatial interpolation", Environmental Pollution, Vol. 157, No. 8–9, 2009, pp. 2378–2385.

[9] Deliang Chen, Tinghai Ou, Lebing Gong, etc, "Spatial interpolation of daily precipitation in China: 1951–2005", Advances in Atmospheric Sciences, Vol. 27, No. 6, 2010, pp.1221-1232.

[10] Owens, J.D.Houston, M.Luebke,ect, "GPU computing," Proceedings of the IEEE, Vol. 96, No.5, 2008, pp.879-899.

[11] D. Blythe, "Rise of the graphics processor," proceedings of the IEEE, Vol. 96, No.5, 2008, pp. 761-778.

[12] Nickolls J., Buck I., Garland M., etc, " Scalable parallel programming with cuda", Queue, Vol.6, No. 2, 2008, pp. 40-53.