Computers, Materials & Continua CMC, vol.60, no.2, pp.599-613, 2019

Satellite Cloud-Derived Wind Inversion Algorithm Using GPU

Lili He[1, 2], Hongtao Bai[1, 2], Dantong Ouyang[1, 2], Changshuai Wang[1, 2], Chong Wang[1, 2, 3] and Yu Jiang[1, 2, *]

Abstract: Cloud-derived wind refers to the wind field data product reversely derived through satellite remote sensing cloud images. Satellite cloud-derived wind inversion has the characteristics of large scale, computationally intensive and long time. The most widely used cloud-derived serial--tracer cloud tracking method is the maximum cross-correlation coefficient (MCC) method. In order to overcome the efficiency bottleneck of the cloud-derived serial MCC algorithm, we proposed a parallel cloud-derived wind inversion algorithm based on GPU framework in this paper, according to the characteristics of independence between each wind vector calculation. In this algorithm, each iteration is considered as a thread of GPU cores, and each thread block array of GPU allocates n*32 threads, and the many thread blocks are allocated to the thread grid. The parameters of the algorithm are passed from CPU to GPU global memory and the storage spaces are previously created on the GPU device before the functions of algorithm are executed. The test results of multiple sets of different inversion models on the NVIDIA Geforce GT and the 4-core 8-thread Core i7-3770 CPU show that the algorithm significantly improves the inversion efficiency. The acceleration ratio is up to 112, and the parallel experiment acceleration ratio is also impressive.

Keywords: Correlation coefficient method, cloud-derived wind, GPU, cross-correlation coefficient, satellite cloud image.

1 Introduction

Cloud-derived wind refers to the wind field data product reversely derived through satellite remote sensing cloud images. A large number of satellite remote sensing image data and terrestrial meteorological information data are needed in the study of global weather and disaster prediction, which can play a positive role in global meteorological disaster forecast as well as protection of personal property [Ouyang (2018); He, Ouyang, Wang et al. (2018); Lompar, Ćurić and Romanic (2017)]. Among them, wind field data is a very important meteorological data in the meteorological field and the acquisition of it has a very important impact on the forecast of aerospace activities and severe weather

[1] College of Computer Science and Technology, Jilin University, Changchun, 130012, China.

[2] Key Laboratory of Symbolic Computation and Knowledge Engineering, Jilin University, Changchun, 130012, China.

[3] Department of Engineering Mechanics, State Marine Technical University of St. Petersburg, St. Petersburg, 190008, Russia.

[*] Corresponding Author: Yu Jiang. Email: jiangyu2011@jlu.edu.cn.

such as typhoon storms. Therefore, it is significant to obtain wind field data [Chen, Chen, Luo et al. (2018); Cervantes, Casanova, Gout et al. (2016)]. In recent years, with the constant improvement of the network of land wind field observation stations, it has become easier for researchers to obtain data on land wind fields. However, compared with the approach of acquiring land wind field data, wind fields in the vast sea, barren and ridiculous high-altitude areas and hot deserts are much poorer. Wind fields are relatively difficult to obtain in these areas. The meteorological satellites operate in the high-altitude orbit around the earth, so it is convenient to collect wind field data in the extreme areas where wind field data is difficult to obtain on the land. The wind can be calculated as long as there is a cloud. Inversion of wind field information in the air based on meteorological satellite remote sensing images has become a very effective means of obtaining short-time interval and high-resolution wind fields, which can compensate for the lack of stations in some environmentally harsh areas at sea and on land [Susanne, Andrey and Miguel (2012); Rich, Frelich, Reich et al. (2016)]. Satheesh Kumar et al. [Satheesh Kumar, Narayana Rao and Taori (2015)] have explored the possibility of implementing an advanced photogrammetric technique, generally employed for satellite measurements, on airglow imager and a very good correspondence was seen between these two wind measurements, both showing similar wind variation. Though the cloud drift wind (CDW) has displayed its good application perspective in numerical weather prediction (NWP), up to the present the CDW data are not actually applied to the daily operation of NWP. Li et al. [Li, Wang, Xue et al. (2008)] have explored the systematic error character of FY-2C CDW and its effects on the initial fields and forecast results of NWP model so as to promote the application of CDW data in operational NWP. Long et al. [Long, Shi and Huang (2000)] has used numerical differentiation, which is developed in recent years to calculate gray gradient, and then realized the inverse cloud motion wind by regularization. At last, through simulation and practical experiments, they compared the wind inverse results between the algorithms with or without gray gradient information when the cloud images include perturbation. The experimental results show that the new algorithm with gradient information can reduce the influence of image disturbance effectively, and also increase the precision of cloud motion winds. We are in a position to find a new way to cloud motion wind inversion.

However, since the satellite remote sensing technology is developing rapidly, the definition of satellite remote sensing cloud images as well as the corresponding resolution is getting much higher. A cloud image contains pixels of millions or even tens of millions. Furthermore, the real-time requirements of products in the meteorological field are getting higher and higher, and the rate of satellite cloud maps is getting faster and faster. If the traditional serial algorithm is used, in most case, the next cloud image is issued before the last cloud image has been processed, and the computational efficiency bottleneck of the cloud-derived wind is gradually revealed. It is getting harder for single machine and single thread to meet current needs of executing the inversion calculation task. In general, the shorter the cloud image time interval, the more tracer clouds used for inversion, the higher compute density of the wind vector, also the higher wind field data quality of the cloud-derived wind, but all of these pose a challenge to the efficiency of the cloud-derived wind inversion algorithm. Although some scholars have proposed a simple algorithm for the cloud-derived wind inversion algorithm, it is difficult to meet the

efficiency requirement of the wind field calculation which is increasingly required for real-time performance. Therefore, in addition to studying the cloud-derived wind inversion algorithm with lower computational cost, the architecture of the multi-core CPU [Jongerius, Anghel, Dittmann et al. (2018)], GPU [Wang, Pan, Davidson et al. (2017)] and cluster [Hulse (2018)] of modern computer is fully utilized to study the parallel cloud-derived wind inversion algorithm, which is also one of the effective ways to improve actual operational efficiency of the inversion algorithm. Wang et al. [Wang, He, Ouyang et al. (2016)] have studied the parallel inversion algorithm of cloud-derived wind based on multi-core CPU and achieved better efficiency improvement, but the number of CPU cores is small, and the performance gain caused by pure multi-core CPU is limited. This paper explores the cloud-derived wind inversion algorithm based on multi-core CPU in order to obtain better performance gain.

2 Basic theory

At present, cloud-derived wind inversion based on satellite cloud image is generally applied to estimate the wind speed and direction by tracking the movement of image blocks in three consecutive time clouds. The height of wind is specified by the height of the corresponding position cloud in cloud map. In actual calculation, cloud-derived wind inversion mainly includes four steps: data preprocessing, tracer cloud tracking, altitude designation and quality control.

The maximum cross-correlation coefficient (MCC) method [Wang, Jia and Cheng (2002)] is the most widely used tracer cloud tracking method, as shown in Fig. 1. The principle can be expressed as: selecting a trace cloud A in the cloud image, and then moving the cloud block A (for the N×N pixel image block) in the target search area S (which is M×M) containing A. Next, calculating all the correlation coefficients of tracking cloud blocks as well as the tracking cloud block A in M. Finally the tracking cloud block B corresponding to the maximum correlation coefficient is the target cloud block.

The cross correlation coefficient R(Δ x, Δ y) is calculated as follows:

$$R(\Delta x,\Delta y)=\frac{\sum_i^N\sum_j^N\left[A(i,j)-\overline{A}\right]\left[B(\Delta y+i,\Delta x+j)-\overline{B}\right]}{\sqrt{\sum_i^N\sum_j^N\left[A(i,j)-\overline{A}\right]^2}\cdot\sqrt{\sum_i^N\sum_j^N\left[B(\Delta y+i,\Delta x+j)-\overline{B}\right]^2}}$$

where $\Delta x,\Delta y\in\left[-\frac{M-N}{2},\frac{M-N}{2}\right]$, $i,j\in[1,N]$, $\Delta x,\Delta y$ are the number of rows and columns of the tracking block deviating from the tracing cloud block respectively. i, j are the index values of the tracing cloud block. A(i, j) and B(i, j) are respectively the pixel gray values of the tracing clouds and the tracking block. $\overline{A}$ and $\overline{B}$ are the average gray levels of the tracking cloud block and the tracking block respectively.

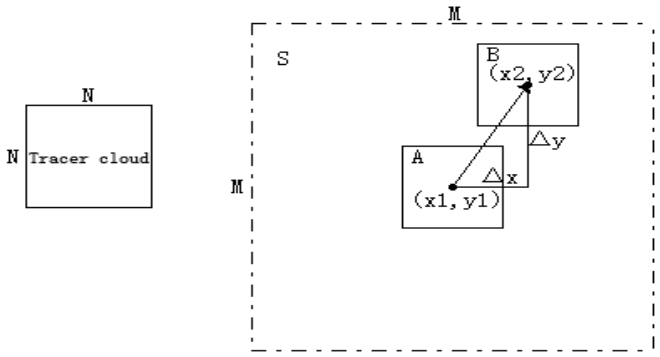

Figure 1: Maximum correlation coefficient method

After obtaining the maximum correlation coefficient, the distances of the tracking cloud block B in the target search area relative to the tracking cloud block A in the x, y directions are Δx and Δy respectively. The latitude and longitude of the center point of the trace cloud block A is (x1, y1) and the latitude and longitude of the center point of the tracking cloud block B is (x2, y2). The earth model (shown in Fig. 2) can be established to calculate the distance d of the target cloud block B relative to the tracking cloud block A. The speed of the wind vector can be obtained by dividing d by the observation time interval t of the two images. The derivation process is as follows:

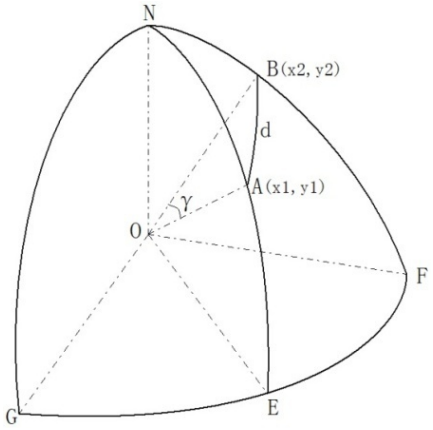

Figure 2: Schematic diagram of wind speed derivation

Wind speed

$$v = \frac{d}{t} \tag{1}$$

The distance between two points A and B (RE is the radius of the Earth)

$$d = RE \times \gamma \tag{2}$$

The spherical angle formed by the two points A and B and the center of the sphere can be calculated by the product of the space vector

$$\cos(\gamma) = \frac{\overrightarrow{OA} \cdot \overrightarrow{OB}}{|\overrightarrow{OA}| \cdot |\overrightarrow{OB}|} \tag{3}$$

Let vector $\overrightarrow{OA}$ be (a1, b1, c1) and vector $\overrightarrow{OB}$ be (a2, b2, c2), then

$$a_1 = RE \cdot \cos(x_1) \cdot \cos(y_1) \quad (4)$$

$$b_1 = RE \cdot \sin(x_1) \cdot \cos(y_1) \quad (5)$$

$$c_1 = RE \cdot \sin(y_1) \quad (6)$$

The same reason

$$a_2 = RE \cdot \cos(x_2) \cdot \cos(y_2) \quad (7)$$

$$b_2 = RE \cdot \sin(x_2) \cdot \cos(y_2) \quad (8)$$

$$c_2 = RE \cdot \sin(y_2) \quad (9)$$

By substituting the formulas (4) to (9) into (3) we can obtain the size of the spherical center angle γ. Then substituting the obtained γ into (2) to obtain the distance d between the two points A and B, and finally substituting (1) with d, the velocity v of the wind vector can be obtained.

The direction of wind vector is calculated using the spherical triangle cosine theorem. As shown in Fig. 3, the plane AED is tangent to the earth spherical surface ABC at point A. A, B are the start and end points of the wind vector respectively.

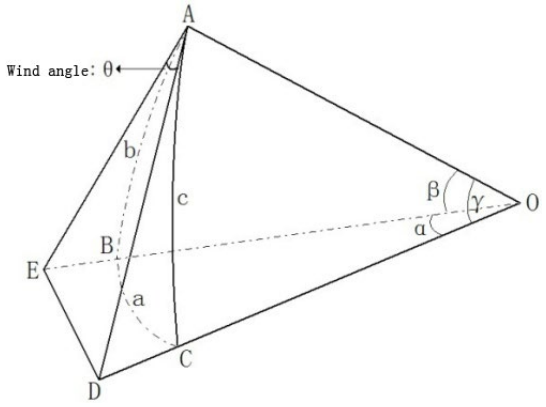

Figure 3: Wind direction derivation

The triangular cosine theorem is applied to the triangle ODE.

$$\cos(\alpha) = \frac{OE^2 + OD^2 - DE^2}{2OE \cdot OD} \quad (10)$$

Since the triangle OAE and the triangle OAD are both right triangles, there are

$$OE^2 = OA^2 + AE^2 \quad (11)$$

$$OD^2 = OA^2 + AD^2 \quad (12)$$

Substituting (11) (12) into (10) and applying the triangular cosine theorem

$$\cos(\alpha) = \cos(\beta) \cdot \cos(\gamma) + \sin(\beta) \cdot \sin(\gamma) \cdot \cos(\theta) \quad (13)$$

Among them

$$\alpha = x_2 - x_1 \quad (14)$$

$$\beta = y_2 - y_1 \quad (15)$$

Substituting (14) and (15) and γ obtained above into (13), the magnitude of θ can be obtained by an inverse sine function. If point A is in the area north of the equator, the wind direction angle is θ. If $x_2 < x_1$, the wind direction angle is 2π-θ. If point A is in the south of the equator, the wind direction angle is π-θ, and if $x_2 < x_1$, the wind direction angle is π+θ.

The height of the wind vector can be specified by the brightness temperature values of the infrared and water vapor channels at the position of the wind vector.

After the wind vector is calculated, the wind speed and direction check are required to eliminate the wind speed and direction deviation in the preliminary calculation results. The cloud maps that define three consecutive time observations are C1, C2, and C3. It is assumed that the wind speed and wind direction of the wind vector V_1 calculated by C1 and C2 are v_1 and θ_1 respectively, and the wind speed and wind direction of the wind vector V_2 are calculated as V_2 and θ_2 by C2 and C3 respectively. The wind direction difference $\theta' = |\theta_1 - \theta_2|$ and the relative difference of wind speed $v' = |\frac{2(v_1 - v_2)}{v_1 + v_2}|$. When the wind direction difference or wind speed difference is greater than a given threshold, the wind vector is removed.

3 Parallel algorithm

3.1 Serial algorithm analysis

Satellite cloud-derived wind algorithm process is mainly divided into the following stages:

1) Data preprocessing

2) Tracer cloud tracking

3) Height specification

4) Quality control

5) Data storage

The satellite data decompression step is to first decompress the satellite cloud image to obtain the data of each channel; the data preprocessing step generally trans-forms the image into a Mercator projection for wind vector inversion, and then image enhancement is needed in order to obtain satellite cloud image data with less noise; At the core of the tracer cloud tracking algorithm, the algorithm uses the maximum correlation coefficient method to perform pixel matching on the center 32*32 area of each block on the cloud image to calculate the size and angle of the wind vector; the height designation part mainly calculates the height of the wind vector. Generally, the calculation method is to obtain temperature based on the gray value of the coordinates of the wind vector, and then the height is derived from the temperature. If the original cloud image is given a temperature calibration table, the height is converted according to the temperature calibration table; the quality control is to check quality according to the wind vectors calculated by the last two of the three consecutive cloud images. The wind vectors with excessive wind speed difference and angle deviation will be removed; the last step is to write the reversed wind to disk using a certain rule.

The serial cloud-derived wind inversion algorithm divides the cloud image of size Width×Height pixels into ((Width×Height))/ρ blocks (ρ is the density of the wind vector),

and uses the wind vector calculation method for each block of the cloud image respectively to calculate wind vectors. In the calculation of each wind vector, the time required to use the MCC method to match the target cloud block is the highest in the calculation time, and the size of the traced image block is mentioned in the related literature [10]. National Satellite Meteorological Center and EUM ETSAT are set to 32×32. If the trace cloud size is used by this standard (i.e., 32×32) and the search area size is set to 64×64, we need 33×33 traces of clouds and the correlation coefficient between the cloud blocks to find the location of the target cloud block. But currently a complete satellite cloud image taken by the meteorological satellite is generally larger (up to megapixel level), which reduces the inversion of all winds.

It can be seen from the above analysis that the main reason for the long wind in-version time in the cloud image in the serial cloud-derived wind inversion algorithm is that it has more iterations than the single iteration time, so the single iteration parallelizes this fine. Granular parallelism is not suitable for cloud-derived wind inversion. Since the calculations of the wind vectors in the cloud map are completely independent, the wind map can be selected for coarse-grained parallelism.

3.2 Parallel algorithm analysis

Using NVIDIA's CUDA architecture, thread is the basic unit of stream processor execution in the GPU, and blocks containing multiple threads are the basic unit of processor scheduling. Multiple threads in the same block use the same memory space, which means they can share access to each other. Since the above block is a basic scheduling unit, it is possible to launch a GPU that has multiple processors, and it is possible to execute more blocks faster, that is, using such a GPU is computationally efficient. The following diagram illustrates that more processors are used, more efficient they perform, as shown in Fig. 4.

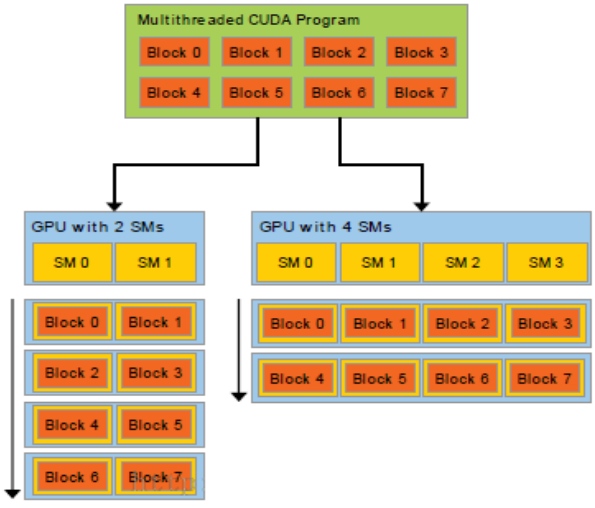

Figure 4: The effect of the number of stream processors on efficiency

The function executed in the stream processor in GPU parallel computing is a kernel function, such as the wind vector inversion task function executed in the stream processing of the GPU, so the function is a kernel function. The kernel function is called by kernel_function<<<M,N>>>(arguments), where M is the size of the thread grid. The variable can be a one-dimensional, two-dimensional or three-dimensional variable, and N represents each block. The number of threads can also use one-dimensional, two-dimensional or three-dimensional variables. In actual use, variables of different dimensions are used according to different scenarios.

Inside the kernel function, CUDA provides some built-in variables for accessing thread indexes, thread blocks and thread grids, etc., such as the following variables:

(1) gridDim: Indicates the size of the grid, gridDim.x, gridDim.y, gridDim.z are the sizes of the x-axis, y-axis, and z-axis respectively.
(2) blockIdx: Indicates the index value of the block in the grid. blockIdx.x and blockIdx.y represent the x-axis index and the y-axis index of the block in the grid respectively.
(3) blockDim: Indicates the size of the block, blockDIm.x and blockDIm.x are the dimensions of the x-axis and the y-axis respectively.
(4) threadIdx: Represents the thread index in the block.

The GPU parallel technology is used to accelerate the cloud-derived wind inversion algorithm, and the MN tasks are dynamically allocated to the GPU stream processors for execution. For the sake of simplicity, the thread application is applied as a one-dimensional variable blocksPerGrid. Each ThreadsPerBlock thread is opened in the thread grid. In this paper, the number of threads on each block is 256, and the number of blocks in the thread grid.

blocksPerGrid=(iMax+threadsPerBlock-1)/threadsPerBlock

The index of each wind vector inversion task to obtain the wind inversion is

windIndex=blockDim.x*blockIdx.x+threadIdx.x

Finally, dev_makeCloud<<<blocksPerGrid, threadsPerBlock>>>(arguments...) is called to allocate and execute each wind inversion task. The threads in each block are executed sequentially in a stream processor and will be reversed. The global wind vector array is opened corresponding to the position of the windIndex.

```
#include <stdio.h>
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include "device_functions.h"
void main(){
//1. Serial part, data reading, preprocessing
//2. Open space for storing cloud image data, correlation coefficient matrix, wind vector array, etc. on GPU devices
//3. GPU parallel part
int threadsPerBlock=N;
int blocksPerGrid=(iMax+threadsPerBlock-1)/threadsPerBlock;
dev_makeCloud<<<blocksPerGrid, threadsPerBlock>>>(Space pointer parameters developed
```

```
on GPU devices
);// Perform wind vector inversion task
//3. Serial part, data summary, data saving
}
```

4 The experimental results

In this section, we compare the calculation results of the serial cloud-derived wind inversion algorithm and parallel algorithm under different parameters, including the calculation accuracy and computational efficiency of the results. The experimental environment is for the CPU to be 4 cores and 8 threads Intel Core i7-3770 3.4 GHZ, GPU NVIDIA Geforce GT650M, and 4G memory. The satellite cloud images used in the experiment are three infrared-cloud images issued by Fengyun-2E satellite at Beijing time on October 11, 2013 at 7:01, 7:31, and 8:01. The size of the cloud map is 2581×1399 pixels, the longitude range is 45° east longitude to 165° east longitude, and the latitude range is 5° south latitude to 60° north latitude.

4.1 Algorithm precision comparison

(1) When the inversion parameters are 22×22 pixels, the tracking area is 44×44 pixels, and the wind density is 2/degree. 10 consecutive wind vectors on the equator are selected as comparison objects. The wind direction accuracy comparison between inversion serial algorithm and parallel algorithm is shown in Tab. 1 (S indicates serial, T-64 represents 64 threads allocated in one thread block, T-128 represents 128 threads in a thread block, and the number of blocks in the thread is the total number of tasks WN divided by the number of threads in a block).

Table 1: Accuracy comparison-1

wind num	longitude (East)	latitude (North)	S		T-64		T-128	
			wind direction (degree)	wind speed (meter/ second)	wind direction (degree)	wind speed (meter/ second)	wind direction (degree)	wind speed (meter/ second)
1	56.50	0.00	122.60	4.00	122.60	4.01	122.60	3.98
2	66.50	0.00	114.72	8.59	114.72	8.57	114.72	8.60
3	67.00	0.00	115.75	8.27	115.75	8.27	115.75	8.27
4	69.50	0.00	110.77	6.08	110.77	6.08	110.77	6.08
5	72.00	0.00	72.39	7.13	72.39	7.14	72.39	7.13
6	72.50	0.00	71.52	11.34	71.52	11.34	71.52	11.33
7	73.00	0.00	70.93	11.00	70.93	10.99	70.93	11.00
8	73.50	0.00	66.34	12.54	66.34	12.54	66.34	12.53
9	74.50	0.00	71.56	11.36	71.56	11.36	71.56	11.36
10	75.00	0.00	78.31	10.64	78.31	10.65	78.31	10.64

(2) The inversion parameters are 32×32 pixels for the tracer. The size of the tracking area is 64×64 and the wind density is 4/degree. The 10 winds with 32 degrees north latitude are selected as the comparison object, parallel and string. The wind vector precision of the row algorithm inversion is shown in Tab. 2. (where S represents serial, T-64 represents 64 threads allocated in one thread block, and T-128 represents 128 thread threads in a thread block, then The number of blocks in the thread grid is the total number of tasks WN divided by the number of threads in each block).

Table 2: Accuracy comparison-2

			S		T-64		T-128	
wind num	longitude (East)	latitude (North)	wind direction (degree)	wind speed (meter/ second)	wind direction (degree)	wind speed (meter/ second)	wind direction (degree)	wind speed (meter/ second)
1	54.00	32.00	186.36	4.68	186.36	4.68	186.36	4.59
2	54.25	32.00	202.28	6.58	202.28	6.59	202.28	6.58
3	54.50	32.00	187.65	6.14	187.65	6.13	187.65	6.15
4	54.75	32.00	204.66	6.70	204.66	6.70	204.66	6.70
5	55.00	32.00	200.98	6.52	200.98	6.52	200.98	6.54
6	55.25	32.00	186.16	6.12	186.16	6.13	186.16	6.12
7	55.50	32.00	209.54	5.34	209.54	5.34	209.54	5.33
8	55.75	32.00	205.07	5.13	205.07	5.12	205.07	5.13
9	56.00	32.00	222.43	4.35	222.43	4.35	222.43	4.34
10	56.25	32.00	217.63	4.05	217.63	4.07	217.63	4.05

As can be seen from Tab. 1 and Tab. 2 above, GPU-based parallel cloud-derived wind inversion, serial inversion and parallel inversion yield most of the same, less partial similar numerical results due to GPU float Point operations also have precision errors, so it is ideal to achieve this result, which proves that the GPU-based parallel algorithm is numerically reliable and efficient.

The cloud-derived product map of the GPU-based parallel inversion algorithm is shown in Fig. 5.

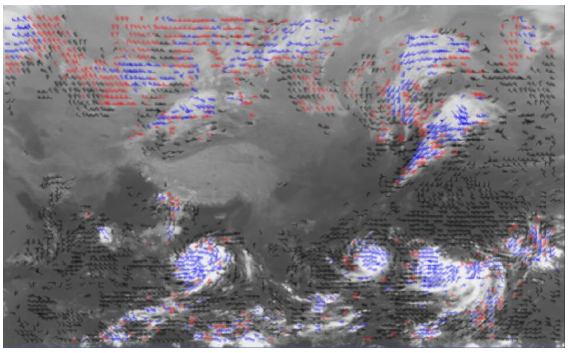

Figure 5: Product map of the parallel algorithm based on GPU

4.2 Performance comparison

The experiment uses the host machine as a 4-core 8-thread CPU. The GPU used is Nvidia' Nvidia GeForce GT650M. The core frequency is 925 MHz, the memory is 1 G, the memory frequency is 5400 MHz, the memory bandwidth is 84.6 GB/s, and the number of stream processors is 768. The theoretical computing power is 1.42 TFLOPs. The satellite cloud image is inverted in the case of different inversion parameters and thread blocks. The relative acceleration ratio and parallel efficiency of the algorithm vary with the inversion parameters and the number of thread blocks as shown in Tab. 3 and Tab. 4.

Table 3: Acceleration ratio and parallel efficiency-1

Inversion Parameter model	Thread block	Execution time (seconds)	Relative acceleration ratio
Tracer cloud:36×36 Tracking area:72×72 Wind density: 4/degree	1(CPU core)	1636.00	--
	1	1401.52	1.17
	2	700.36	2.34
	4	349.78	4.68
	8	174.64	9.37
	16	87.44	18.71
	24	65.59	24.94
	32	43.77	37.38
	40	44.55	36.72
	48	33.39	49.00
	56	33.33	49.08
	64	22.55	72.55
	72	24.69	66.26

	80	23.55	69.47
	88	23.52	69.56
	96	23.47	69.71
	104	24.98	65.49
	112	24.83	65.89
	120	24.83	65.89
	128	14.14	115.70
	136	24.48	66.83
	144	24.48	66.83
	152	24.47	66.86
	160	24.47	66.86
	168	24.5	66.78
	176	24.52	66.72
	184	24.5	66.78
	192	24.5	66.78
	200	24.86	65.81
	208	24.86	65.81
	216	24.83	65.89
	224	14.55	112.44
	232	24.88	65.76
	240	24.86	65.81
	248	14.52	112.67
	256	14.52	112.67

Table 4: Acceleration ratio and parallel efficiency-2

Inversion Parameter	Thread block	Execution time (seconds)	Relative acceleration ratio
Tracer cloud: 36×36 Tracking area: 72×72 Wind density: 2/degree	1(CPU core)	1205.06	--
	1	1086.94	1.11
	2	542.72	2.22
	4	270.61	4.45
	8	139.67	8.63
	16	96.05	12.55
	24	69.86	17.25

32	52.41	22.99
40	34.95	34.48
48	35.59	33.86
56	26.64	45.23
64	26.63	45.25
72	17.95	67.13
80	19.73	61.08
88	18.78	64.17
96	18.78	64.17
104	18.78	64.17
112	20	60.25
120	19.88	60.62
128	19.88	60.62
136	11.34	106.27
144	19.59	61.51
152	19.59	61.51
160	19.58	61.55
168	19.59	61.51
176	19.61	61.45
184	19.59	61.51
192	19.59	61.51
200	19.88	60.62
208	19.88	60.62
216	19.84	60.74
224	11.63	103.62
232	19.89	60.59
240	19.89	60.59
248	11.61	103.80
256	11.61	103.80

It can be seen from the analysis of Tab. 3 and Tab. 4 that in the case where the inversion parameters are the same, the more thread blocks are basically followed, the larger acceleration ratio is. When the thread block is smaller than 64, the parallel acceleration ratio is increased by 2 times. When it is greater than 64, the parallel acceleration ratio is related to the division of the thread block. The highest acceleration ratio of the two experiments reaches 112 and 103 and the acceleration effect is very significant.

5 Conclusion

This paper discusses the parallel computing problem of cloud-derived wind inversion. According to the characteristics of cloud-derived wind inversion, a parallel algorithm based on GPU for many-core computing is designed and implemented. By analyzing the calculation results, it is known that assigning reasonable parallel granularity can guarantee the correctness of the calculation results and effectively improve, compared with the serial algorithm, the calculation efficiency. Compared with the serial algorithm, it provides an efficient calculation for large-scale, short-interval and high-density wind vector.

In recent years, the application of cluster computing is becoming much more extensive, and the mixed using multi-core CPU, MPI and GPUs [Pawliczek, Dzwinel and Yuen (2014)] mixed programming mode for design algorithm design will also be the direction to further improvement of the efficiency of cloud-derived wind cloud windward inversion. In addition to the application of parallel technology, the efficiency improvement of the cloud-derived wind inversion algorithm is also worth considering from the perspective of optimizing the algorithm of the wind vector inversion. For example, the other tracer cloud tracking algorithms and the optimization of thee tracking efficiency are considered.

Acknowledgements: This work was supported in part by the National Natural Science Foundation of China (61872160, 51679105, 51809112, 61672261).

References

Cervantes, D. A.; Casanova, P. G.; Gout, C.; Moreles, A. M. (2016): A line search algorithm for wind field adjustment with incomplete data and RBF approximation. *Computational & Applied Mathematics*, vol. 37, no. 3, pp. 2519-2532.

Chen, Y.; Chen, C. L.; Luo, X.; Zhang, Y.; Yang, Z. H. et al. (2018): Research on wind field algorithm of wind lidar based on BP neural network and grey prediction. *International Conference on Optical Instruments & Technology: Advanced Laser Technology & Applications*, vol. 10619.

He, L. L.; Ouyang, D. T.; Wang, M.; Bai, H. T.; Yang, Q. L. et al. (2018): A method of identifying thunderstorm clouds in satellite cloud image based on clustering. *Computers, Materials & Continua*, vol. 57, no. 3, pp. 549-570.

Hulse, P. (2018): Review: Beowulf cluster computing with Linux, second edition. *Computer Journal*, vol. 48, no. 3, pp. 379-380.

Jongerius, R.; Anghel, A.; Dittmann, G.; Mariani, G.; Vermij, E. et al. (2018): Analytic multi-core processor model for fast design-space exploration. *IEEE Transactions on Computers*, vol. 67, no. 99, pp. 755-770.

Li, H. H.; Wang, M.; Xue, J. S.; Qi, M. H. (2008): A study on the application of FY-2C cloud drift wind in the mesoscale numerical Model. *Acta Meteorological Siica*, vol. 66, no. 1, pp. 50-58.

Lompar, M.; Ćurić, M.; Romanic, D. (2017): Simulation of a severe convective storm using a numerical model with explicitly incorporated aerosols. *Atmospheric Research*, vol. 194, pp. 164-177.

Long, Z. Y.; Shi, H. Q.; Huang, S. X. (2011): A new idea of cloud motion wind derived from satellite images. *Acta Physica Sinica*, vol. 60, no. 5, pp. 840-845.

Ouyang, H. T. (2018): Input optimization of ANFIS typhoon inundation forecast models using a multi-objective genetic algorithm. *Journal of Hydro-Environment Research*, vol. 19, pp. 16-27.

Pawliczek, P.; Dzwinel, W.; Yuen, D. A. (2014): Visual exploration of data by using multidimensional scaling on multicore CPU, GPU, and MPI cluster. *Concurrency & Computation Practice & Experience*, vol. 26, no. 3, pp. 662-682.

Rich, R. L.; Frelich, L.; Reich, P. B.; Bauer, E. M. (2016): Detecting wind disturbance severity and canopy heterogeneity in boreal forest by coupling high-spatial resolution satellite imagery and field data. *Remote Sensing of Environment*, vol. 114, no. 2, pp. 299-308.

Satheesh Kumar, S.; Narayana Rao, T.; Taori, A. (2015): A novel approach for the extraction of cloud motion vectors using airglow imager measurements. *Atmospheric Measurement Techniques*, vol. 8, no. 3, pp. 2657-2682.

Susanne, L.; Andrey, P.; Miguel, B. (2012): High-resolution satellite measurements of coastal wind field and sea state. *International Journal of Remote Sensing*, vol. 3, no. 23, pp. 7337-7360.

Wang, Y.; Pan, Y.; Davidson, A.; Wu, Y. D.; Yang, C. et al. (2017): Gunrock: GPU graph analytics. *ACM Transactions on Parallel Computing*, vol. 9, no. 4, pp. 3.

Wang, C. S.; He, L. L.; Ouyang, D. T.; Bai, H. T. (2016): Satellite cloud drift winds parallel inversion algorithm based on multi-core CPU. *Journal of Jilin University: Science Edition*, vol. 54, no. 3, pp. 539-546.

Wang, Y. B.; Jia, X. Z.; Cheng, J. (2002): A numerical differentiation method and its application to reconstruction of discontinuity. *Inverse Problems*, vol. 18, no. 6, pp. 1461-1476.