A Revised Satellite Cloud-Derived Wind Inversion Algorithm Based on Computer Cluster

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

Abstract: In view of the satellite cloud-derived wind inversion has the characteristics of large scale, intensive computing and time-consuming serial inversion algorithm is very difficult to break through the bottleneck of efficiency. We proposed a parallel acceleration scheme of cloud-derived wind inversion algorithm based on MPI cluster parallel technique in this paper. The divide-and-conquer idea, assigning winds vector inversion tasks to each computing unit, is identified according to a certain strategy. Each computing unit executes the assigned tasks in parallel, namely divide-and-rule the inversion task, so as to reduce the efficiency bottleneck of long inversion time caused by serial time accumulation. In the scheme of parallel acceleration based on MPI cluster, an algorithm based on performance prediction is proposed to effectively implement load balance of MPI clusters. Through the comparative analysis of experiment data using the parallel scheme of this parallel technology framework, it shows that this parallel technology has a certain acceleration effect on the cloud-derived wind inversion algorithm. The speedup of the MPI-based parallel algorithm reaches 14.96, which achieved the expected estimate. At the same time, this paper also proposes an efficiency optimization algorithm for cloud-derived wind inversion. In the case that the inversion of wind vector accuracy loss is minimal, the optimized algorithm execution time can be up to 13 times faster.

Keywords: Cloud-derived wind, wind vector, inversion, parallel computing, MPI, speedup ratio.

1 Introduction

Cloud-derived wind refers to the wind field data product reversely derived from satellite remote sensing cloud images. A great number of terrestrial meteorological information data and satellite remote sensing image data can play an important role in global meteorological disaster forecast and protection of personal property in the study of global

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

[2] Symbol Computation and Knowledge Engineer of Ministry of Education, Jilin University, Changchun, 130012, China.

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

[*] Corresponding Author: Hongtao Bai. Email: baiht@jlu.edu.cn.

Received: 22 January 2020; Accepted: 30 March 2020.

weather as well as disaster prediction, see Lompar et al. [Lompar, Ćurić and Romanic (2017); Li, Wang, Xue et al. (2008)]. Among them, wind field data has a great impact on the forecast of aerospace activities and severe weather, for which it is significant to obtain wind field data [Cervantes, Casanova, Gout et al. (2016); Chen, Chen, Luo et al. (2018)]. In recent years, since the network of land wind observation stations are improving constantly, it has become easier for researchers to obtain data on land wind fields. However, wind fields are relatively difficult to obtain in the barren, vast sea, hot deserts and high-altitude areas. The meteorological satellites operate in the high-altitude orbit around the earth so it can obtain wind field data in the extreme areas. Inversion of wind field information in the air based on meteorological satellite remote sensing images has become a very effective means of building 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 [Ouyang (2018); Rich, Frelich, Reich et al. (2016)].

However, since the satellite remote sensing technology is developing rapidly, the definition as well as the corresponding resolution of satellite remote sensing cloud images is becoming much higher. 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. The computational efficiency bottleneck of the cloud-derived wind is revealed gradually. We have promoted two novel parallel cloud-derived algorithms based on multi-core CPU and GPU [Wang, He, Ouyang et al. (2016); He, Bai, Ouyang et al. (2019)], however, it is still harder for single machine 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 computing density of the wind vector, also the higher wind field quality of the cloud-derived wind. However, all of these pose a challenge to the efficiency of the cloud-derived wind inversion algorithm. Therefore, in addition to studying the cloud-derived wind inversion algorithm with lower computational cost, the architecture of the 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.

Computer cluster is a relatively new technology and one of the core technologies in computer technology. Here, a cluster refers to a group of computers that are independent of each other and interconnected by a high-speed network [Zhang (2015)]. A hierarchical clustering technology under Map Reduce environment is utilized to process the experiment data of the tailings of rare earth ion in Ren et al. [Ren, Wang, Feng et al. (2019)], it researched the leaching trend of rare earth ions in tailings under optimum conditions according to analyze of the experiment data.

This paper proposes a parallel acceleration scheme for cloud-derived wind inversion algorithm based on MPI cluster parallel technology.

2 Analysis of serial algorithm

Satellite cloud-derived wind algorithm is divided into five stages:

(1) Data preprocessing

(2) Tracer cloud tracking

(3) Height specification

(4) Quality control

(5) Data storage

The step of satellite data decompression first decompresses the satellite cloud image to obtain the data of each channel; the data preprocessing step generally transforms 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. In the height designation part the calculation method is used to obtain temperature based on the gray value of the coordinates of the wind vector. Then the height is derived from the temperature; the quality control is to check the 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 by using a certain rule [He, Bai, Ouyang et al. (2019)].

The detailed process of this algorithm using natural language description can be divided into the following steps:

(1) Decompress the satellite cloud image data sent by the satellite;

(2) Take three satellite infrared channel cloud images with a time interval of Dt, named as cloud map 1, cloud map 2, and cloud map 3 in order of time;

(3) The image gray enhancement algorithm performs gray enhancement of the three images separately to track the target cloud more accurately;

(4) Divide the cloud images into (Width/ρ)×(Height/ρ) tracking blocks and ρ is the inversion density of wind vector;

(5) Traverse the tracking blocks of the cloud in turn, from top to bottom, from left to right until the end of the traversal, then the entire program ends;

(6) Position the 32×32 rectangular pixel area A at the center position in the current Block, and mark the center point as (x1, y1);

(7) Calculate the gray scale variance and standard deviation of A region as fSub1;

(8) Create a (32+1)×(32+1) correlation coefficient storage matrix Array;

(9) From the first 32×32 pixel area in the upper left corner of the same block Block in the cloud image 2, traverse from top to 0, down from left to right, until the end of the traversal, return to Step (3). This block is named as B;

(10) Calculate the variance and standard deviation fSubB of the B block, then calculate the correlation coefficient pR between blocks A and B and write it into Array[x][y];

(11) Find the largest correlation coefficient pRMax in the correlation coefficient matrix Array, and the corresponding position is (x2, y2);

(12) According to the coordinates (x1, y1) and (x2, y2), the wind vector distance and angle of the block Block can be calculated. By dividing the distance of the wind vector at the time interval Dt, the wind speed can be obtained;

(13) Using cloud images 2 and 3, the wind vector calculated by the same algorithm performs quality verification on the wind vector calculated this time, and rounds off the wind vector with a large error;

(14) According to the gray scale information of the point (x1, x2), the height of the wind vector can be calculated;

(15) Finally, write the obtained wind speed, angle, height and other information to the disk in a specified format;

(16) Return to Step (9);

3 Variable step size improvement

In the cloud-derived wind inversion algorithm, the tracking of the tracer cloud is the most critical step, and it is also the most time-consuming step. Thus the importance of optimizing this part is self-evident. The tracking algorithm used in this paper is the maximum correlation coefficient method.

The correlation coefficient is a statistical indicator to measure the correlation between two variables. The calculations are usually performed by the difference method, based on the dispersion of their respective average values, and the degree of correlation is reflected. The concept of the maximum correlation coefficient method is easy to see, that is, calculating the degree of correlation between a variable A and a plurality of variables, and the variable having the greatest correlation with A is selected as the output of the algorithm. In this paper, the scene to use the maximum correlation coefficient method is to find the 32×32 pixel region with the largest similarity between the 32×32 pixel regions in the center of the pixel block in which the wind vector is to be inverted. The relationship between similarity and correlation coefficient is monotonically increasing. Thus, by using this algorithm we can find the position of the target block and then calculate the relevant values of the wind vector. The schematic diagram is shown in Fig. 1.

In this algorithm, the correlation coefficient between two blocks is calculated, and the calculation of the correlation coefficient array takes a long time. If the size of the block is N×N and then the tracking area where the block is located is M×M. If every correlation coefficient is calculated, then (M-N+1)×(M-N+1) correlation coefficients need to be calculated. The time is more than a thousand times of calculating the correlation coefficient. Therefore, in this paper we will optimize the calculated quantity.

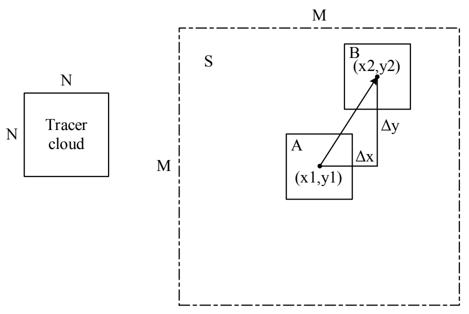

Figure 1: Schematic diagram of tracer cloud tracking

In this section we propose a concept-variable step size iterative calculation. Variable step size search method means that in tracking area S, the moving step size of tracing cloud block is changed from the original 1 strategy to the variable step size movement. We choose a larger value as the initial step size of the search, then gradually reduce the step size while narrowing the search range until the step size becomes 1. For each step size, the best target matching cloud block in the current step size is found and when the length becomes 1, the approximate global best matching cloud block B is found. If the initial step size is 8, then the following derivation process is available.

The first movement step size is 8, and then (M/8)×(M/8) correlation coefficient values are calculated. After finding the biggest of these correlation coefficient values, we can find the N×N block A1 with the highest similarity. The method is shown in Fig. 2.

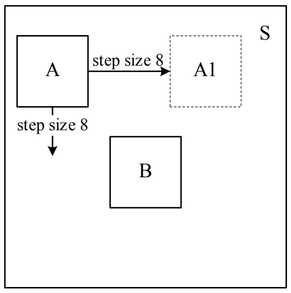

Figure 2: Iterative search with step size of 8

After finding A1, in order to avoid errors, we increase the side length of the A1 block by step pixels, and then reduce the step size to 8/2 or 4 pixel points, that is, step is 4. Search the N×N region with the largest correlation coefficient in the region of N+step side length centered on A1, and then the correlation coefficients of each N×N block and B block are calculated respectively. The coefficient matrix size is (N+step)/4×(N+step)/4, i.e., (N+step) 2/16, which means that by iteratively searching for these steps, we can find the largest correlation coefficient block in the case of Step 4. The search algorithm is shown in Fig. 3.

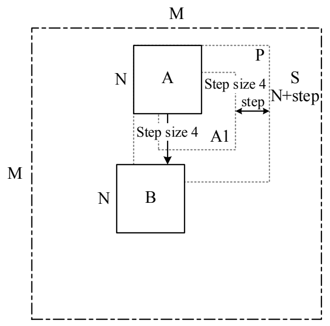

Figure 3: Iterative search with step size of 4

According to this algorithm, step is gradually reduced by half. Finally when the step size is 1, the largest correlation coefficient block Block is the block with the highest similarity with the central block B, so according to the position information of two blocks, we can calculate the speed and angle of wind vector.

We can calculate how many times of iterative search are needed to find the best similar block by using the original method of tracing cloud tracking

StepCount1=(M-N+1)2 (1)

After algorithm optimization, the number of iterative searches that need to find the best similar block is

StepCount2=(M/step)2+step/2+step/4+…+1 (2)

In the case that the step size is modified to M, 64, N, 32, and the Step, 8, the original method requires 1089 steps of search, and the optimized method requires 71 steps, which greatly reduces the time required for search, and it can theoretically accelerate by more than 15 times. However, using this method may result in a certain error in the inversion of wind vector, and the subsequent precision optimization can make adjustments related to the M size for the range of the search again after the variable step size.

4 Parallel algorithm design

4.1 Parallel idea

It can be seen from the in-depth analysis of the steps of the serial algorithm in Chapter 3 that the loop of the wind vector inversion according to the wind vector index in Step (5) is the most time-consuming part of the whole algorithm. However, after analysis we find that there is almost no data dependency before the calculation of each cycle except the index number of the wind vector, that is, each calculation has relative independence.

According to the above analysis, we can start from the loop of parallel Step (5). This step can be parallelized by Multi-core of CPU, GPUs or Multi-computers. While, there is an extensive method, using many computers, not one CPU or GPU in single unit. We processed it using MPI cluster computing technology. Further analysis shows that the loop of tracer cloud tracking in Step (9) is the most time-consuming part of the loop, and it also has the characteristics of no data dependency before each loop calculation, but this paper only applies coarse-grained parallel optimization to the original serial algorithm, so parallel processing of Step (9) loops is not considered.

The cloud-derived wind inversion serial algorithm divides the cloud image of 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. In the calculation of each wind vector, the time required to use the MCC method to match the target cloud block is the longest in the whole calculation time, and the size of the traced image block is mentioned in the related literature [Susanne, Andrey and Miguel (2012)]. National Satellite Meteorological Center and EUM ETSAT set the size to 32×32. If the tracer cloud size uses this standard (i.e., 32×32) and the search area size is set to 64×64, 33×33 correlation coefficients between tracing cloud blocks and tracking cloud blocks are required to find the location of the target cloud block. However, currently a complete satellite cloud image taken by the meteorological satellite is generally larger (up to megapixel level), which reduces the computational efficiency of inversion.

It can be seen from the above analysis that the main reason for the long wind inversion time of the wind vectors in the serial cloud-derived wind inversion algorithm is that it has more iterations rather than longer single iteration time. Thus, granular parallelism is not suitable for cloud-derived wind inversion. Since the calculations of the wind vectors in the cloud images are completely independent, coarse-grained parallelism can be selected to calculate the wind vectors. The algorithm idea can be basically described as the following process:

(1) Decompress the satellite cloud image data sent by the satellite;

(2) Take three satellite infrared channel cloud images with a time interval of Dt, named as cloud map 1, cloud map 2, and cloud map 3 in order of time;

(3) The image gray enhancement algorithm performs gray enhancement of the three images separately in order to track the target cloud more accurately;

(4) Divide the cloud image into (Width/ρ)×(Height/ρ) tracking blocks and ρ is the inversion density of the wind vector;

(5) Allocate a specified calculation processing unit to all blocks, using a certain rule for task assignment of n parallel processing units, and the tasks in each parallel processing unit represent a complete wind vector inversion task;

(6) Start n parallel transactions to perform parallel computation on the assigned processing tasks.

(7) Data summary. The wind vectors of all parallel processing unit inversions are summed together and valid wind vector data is written to disk for storage.

(8) The program ends.

The basic flow chart of the parallel cloud-derived wind inversion algorithm is shown in Fig. 4.

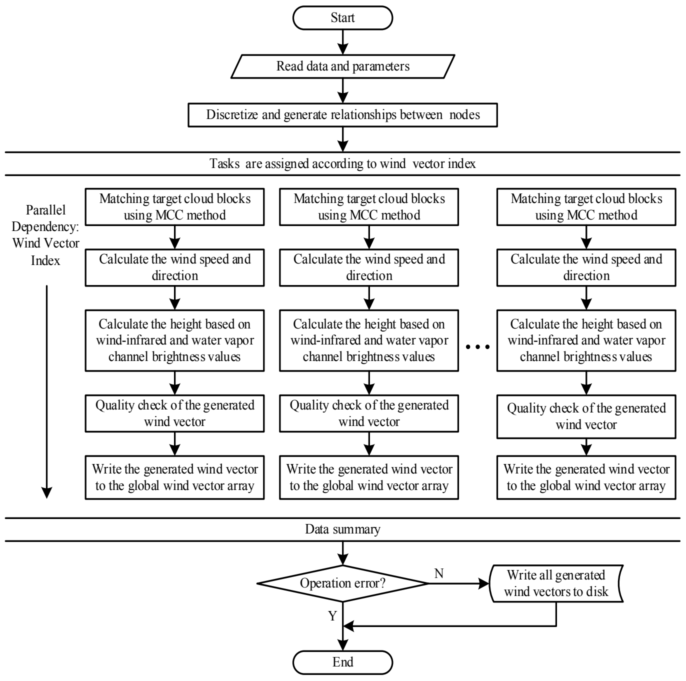
Start
Read data and parameters
Discretize and generate relationships between nodes
Tasks are assigned according to wind vector index
Parallel Dependency: Wind Vector Index
Matching target cloud blocks using MCC method
Calculate the wind speed and direction
Calculate the height based on wind-infrared and water vapor channel brightness values
Quality check of the generated wind vector
Write the generated wind vector to the global wind vector array
...
Data summary
Operation error?
N
Y
Write all generated wind vectors to disk
End

Figure 4: Parallel flow chart

4.2 Parallel algorithm description

Parallel programming based on MPI messaging mechanism has two design modes: peer-to-peer mode and master-slave mode. The normal MPI program uses the master-slave mode. The master machine is responsible for dynamically assigning tasks and data summary. The master does not participate in the task execution and only the slave handles the wind vector inversion task. After the slave machine has finished the task, it sends a message to the master machine for requesting a new task. The experiment in this paper uses the master-slave mode for task assignment and processing.

Since the network data transmission is time-consuming, which will cause the machine which has finished the task to run idle as well as causing machine waste, the network transmission cost should be minimized. Therefore, the task assignment in this paper is only done at the beginning, and it is not dynamically allocated during the task execution process, which can greatly reduce the network overhead and ensure that there are as few idling situations as possible in each process of the MPI.

However, MPI parallelism usually uses the cluster computing mode. In the cluster, there are different performances of different machines and different throughputs of task execution, which results in the tasks not being evenly distributed. In this paper we use a common performance prediction method for the cluster of single machines. Before performing the cloud-derived wind inversion task, we execute a performance measurement study for all the processes of all the machines in the cluster, and the time required for each process to execute this example is TPC (time of parallel calculated). The time for the execution of each process is summarized, and then all the wind vector inversion tasks are weighted and distributed to each machine in the cluster according to the time returned by all machines. The weighted task quantity is calculated as follows:

(1) Record the tpc of each performance measurement example, and summarize it into the array tpc;

(2) Calculate the weight of the tpc of each process for the total task duration w [i], w [i]=tpc [i]/sum (tpc [1, 2, ..., n]);

(3) According to the calculated weight of each process, assign the number of tasks of each process, noftask [i]=w [i]×WN, (WN is the total number of tasks);

(4) Assign tasks to n processes in sequence, for example, assigns noftask [1] tasks to process 1, and so on assigns noftask [n] tasks to process n until all tasks are assigned.

According to the MPI task scheduling method described above, the cloud-derived wind inversion algorithm based on the MPI framework can be described as

(1) Cloud image data preprocessing

(2) Assign performance measurement examples to all processes in the cluster and execute them in parallel;

(3) Summarize tpc and calculate the number of tasks assigned to each process;

(4) Assign all the cloud-derived wind inversion tasks to all processes in the cluster, according to the distribution scheme calculated in Step (2);

(5) Perform the wind vector inversion task in parallel;

(6) Summarize the wind vectors of all process inversions and write them to disk.

The corresponding parallel scheme flow chart is shown in Fig. 5.

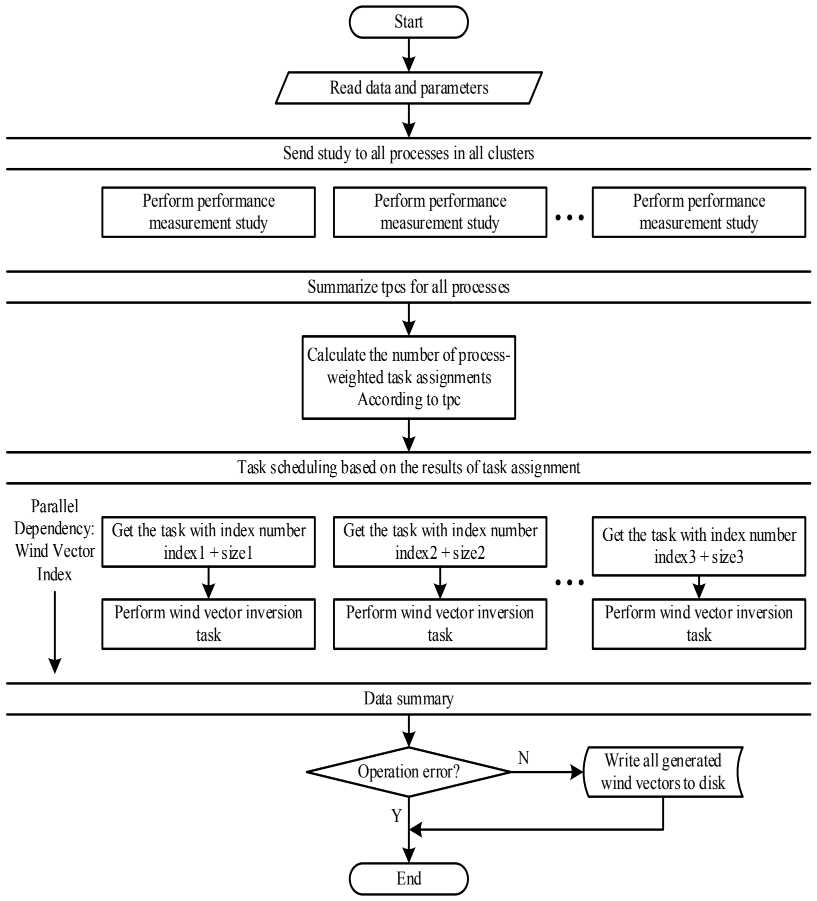
Start
Read data and parameters
Send study to all processes in all clusters
Perform performance measurement study
Perform performance measurement study
Perform performance measurement study
Summarize tpcs for all processes
Calculate the number of process-weighted task assignments According to tpc
Task scheduling based on the results of task assignment
Parallel Dependency: Wind Vector Index
Get the task with index number index1 + size1
Get the task with index number index2 + size2
Get the task with index number index3 + size3
Perform wind vector inversion task
Perform wind vector inversion task
Perform wind vector inversion task
Data summary
Operation error?
N
Y
Write all generated wind vectors to disk
End

Figure 5: Parallel flow chart

5 Experimental test

5.1 Variable step size improvement

5.1.1 Algorithm precision comparison

When the inversion parameter is a tracing cloud block size of 32×32 pixels, the tracking area size is 64×64 pixels, and the wind vector density is 1/degree. After the experiment, 2593 wind vectors are inversely performed before the algorithm optimization and 2546 wind vectors are inversely performed after the algorithm optimization. In this section, 10 consecutive wind vectors with different latitudes and longitudes are randomly selected as the comparison objects. The wind vector accuracy of the inversion before and after the algorithm optimization is shown in Tab. 1 (where OB stands for algorithm optimization before experiment and OA stands for algorithm optimization after experiment).

Table 1: Wind vector accuracy comparison

			OB		OA	
Wind Num	Longitude (East)	Latitude (North)	Wind Direction (Degree)	Wind Speed (Meter/ Second)	Wind Direction (Degree)	Wind Speed (Meter/ Second)
1	51.00	-3.00	108.60	4.62	108.60	4.62
2	108.00	13.00	343.00	4.67	343.00	4.67
3	61.00	14.00	243.55	4.47	243.55	4.47
4	126.00	19.00	98.20	8.48	98.20	8.48
5	142.00	22.00	93.68	7.09	93.68	7.09
6	147.00	23.00	100.26	6.52	100.26	6.52
7	133.00	38.00	246.13	7.76	246.13	7.76
8	88.00	47.00	256.66	16.25	--	--
9	106.00	52.00	250.85	4.67	250.85	4.67
10	81.00	57.00	278.70	4.52	278.70	4.52

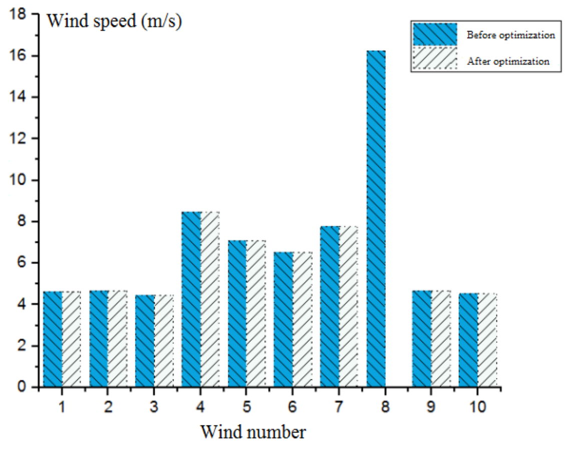

Figure 6: Accuracy comparison of randomly selected wind speeds

It can be seen from Fig. 6 that before and after the optimization of the cloud-derived wind inversion algorithm, except for the calculation missing of the eighth wind vector optimization algorithm, other wind vector speeds and angles obtain the same numerical results. This can prove that the loss of the inversion accuracy of the optimized cloud-derived wind inversion algorithm is not too great.

5.1.2 Experiment comparison of algorithm efficiency

In the experiments, we use a 4-core 8-thread CPU to invert the satellite cloud image with different inversion parameters and number of threads. The inversion execution time before and after the algorithm optimization and the acceleration multiplier of the optimized algorithm after the optimization algorithm are as follows.

Table 2: Efficiency comparison

Inversion Parameter Model	Thread Block	Execution Time Before Optimization (Seconds)	Execution Time After Optimization (Seconds)	Relative Acceleration Ratio
Tracer Cloud: 36×36 Tracking Area: 72×72 Wind Density: 4/Degree	1	1636.00	123.38	13.26
	2	842.90	73.04	11.54
	3	603.18	53.38	11.30
	4	487.54	44.08	11.06
	5	426.76	37.77	11.30
	6	388.14	35.71	10.87
	7	372.62	34.34	10.85
	8	357.50	33.10	10.80
Tracer Cloud:36×36 Tracking Area:72×72 Wind Density: 2/Degree	1	405.76	36.07	11.25
	2	213.32	21.27	10.03
	3	152.07	15.19	10.01
	4	123.86	12.84	9.65
	5	107.04	10.94	9.78
	6	97.04	10.11	9.60
	7	92.32	9.60	9.62
	8	89.40	9.39	9.52
Tracer Cloud: 36×36 Tracking Area: 72×72 Wind Density: 1/Degree	1	102.54	10.27	9.98
	2	52.81	6.11	8.65
	3	38.39	4.45	8.62
	4	31.36	3.67	8.54
	5	27.09	3.29	8.23
	6	24.90	3.02	8.25
	7	23.87	2.94	8.12
	8	22.65	2.76	8.21

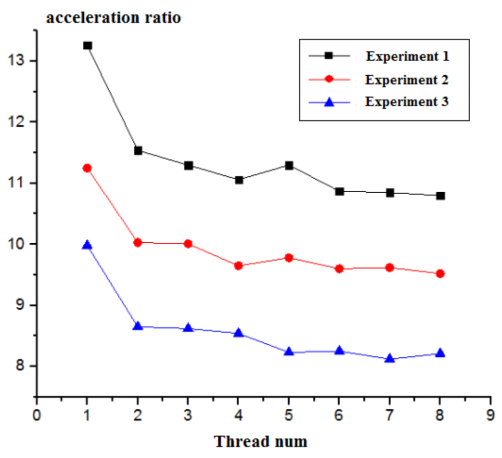

Figure 7: Acceleration multiple of optimized algorithm under different models

It can be seen from the analysis of Tab. 2 and Fig. 7 that the efficiency of cloud-derived wind inversion is significantly improved after the algorithm is optimized. The acceleration multiplier is at least 8 times and the highest is 13 times or more, which shows that the algorithm after optimization is more efficient.

5.2 Parallel algorithm

5.2.1 Algorithm precision comparison

When the inversion parameter is a tracing cloud block size of 22×22 pixels, the tracking area is 44×44 pixels, and the wind vector density is 2/degree, 10 consecutive wind vectors on the equator are selected as comparison objects. The comparison of wind vector accuracy between Parallel and Serial algorithm Inversion is shown in Tab. 3 (where S represents serial, P-2 represents 2 parallel PC, P-4 represents 4 parallel PC).

Table 3: Accuracy comparison

			S		P-2		P-4	
wind num	latitude (North)	longitude (East)	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	4.00	122.60	4.00
2	66.50	0.00	114.72	8.59	114.72	8.59	114.72	8.59
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	7.13	72.39	7.13
6	72.50	0.00	71.52	11.34	71.52	11.34	71.52	11.34

7	73.00	0.00	70.93	11.00	70.93	11.00	70.93	11.00
8	73.50	0.00	66.34	12.54	66.34	12.54	66.34	12.54
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	10.64	78.31	10.64

As seen from Figs. 6 and 7, in the process of cloud-derived wind inversion, serial inversion and parallel inversion obtain identical numerical results, which proves that the parallel method based on MPI is numerically credible and effective.

The cloud-derived wind product image based on the MPI-based parallel inversion algorithm is shown in Fig. 8.

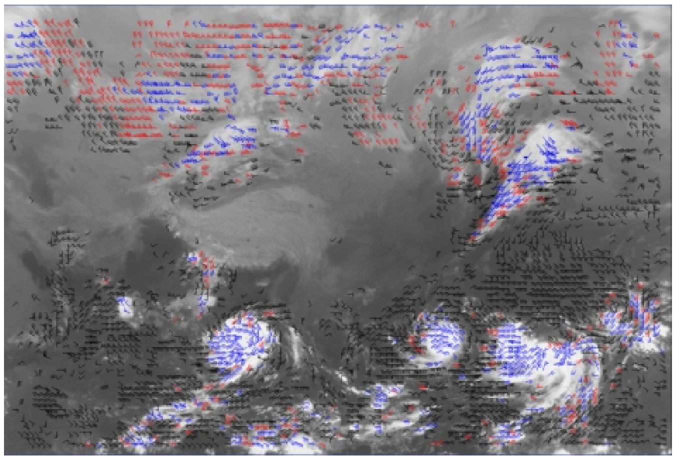

Figure 8: Product map of the parallel algorithm based on MPI

5.2.2 Experimental comparison of algorithm efficiency

In this experiment we use five dual-core Core CPUs and one 4-core i5 processor, and the sixth of which is a 4-core CPU. The CPUs invert the satellite cloud image respectively with different inversion parameters and number of PCs (the number of PCs actually is involved in the calculation here).

The relative acceleration ratio and parallel efficiency without using the load-balancing algorithm vary with the inversion parameters and the number of PCs as shown in Tab. 4. (The serial algorithm is executed on a dual-core Core CPU)

Table 4: Acceleration ratio and parallel efficiency (non-load balancing)

Inversion Parameter Model	PC/ Process	Execution Time (Seconds)	Relative Acceleration Ratio	Parallel Efficiency
Tracer Cloud: 36×36 Tracking Area: 72×72	1	3828.24	--	--
	2	2004.30	1.91	95.50%
	3	1558.65	2.89	96.33%

Wind Density: 4/Degree	4	1020.87	3.75	93.75%
	5	823.28	4.65	93.00%
	6	719.59	5.23	87.16%
Tracer Cloud: 36×36 Tracking Area: 72×72 Wind Density: 2/Degree	1	949.47	--	--
	2	494.51	1.92	96.00%
	3	331.97	2.86	95.33%
	4	260.83	3.64	91.00%
	5	199.88	4.75	95.00%
	6	163.70	5.80	96.67%
Tracer Cloud: 36×36 Tracking Area: 72×72 Wind Density: 1/Degree	1	239.94	--	--
	2	123.57	1.95	97.50%
	3	88.19	2.72	90.67%
	4	62.31	3.85	96.25%
	5	51.03	4.70	94.00%
	6	43.47	5.52	92.00%
Tracer Cloud: 36×36 Tracking Area: 64×64 Wind Density: 1/Degree	1	145.29	--	--
	2	77.61	1.89	94.50%
	3	54.21	2.68	89.33%
	4	37.74	3.85	96.25%
	5	31.10	4.67	93.40%
	6	25.57	5.68	94.67%
Tracer Cloud: 32×32 Tracking Area64×64 Wind Density: 2/Degree	1	585	--	--
	2	312.80	1.85	92.50%
	3	210.43	2.78	92.67%
	4	150.39	3.89	97.25%
	5	126.36	4.63	92.60%
	6	100.17	5.84	97.33%
Tracer Cloud: 32×32 Tracking Area64×64 Wind Density: 4/Degree	1	2325.46	--	--
	2	1228.05	1.89	94.50%
	3	836.50	2.78	92.67%
	4	593.23	3.92	98.00%
	5	495.84	4.69	93.80%
	6	444.64	5.23	87.17%

By analyzing the data in the above table, it can be seen that, in the case where the inversion parameters are the same, the larger the number of PCs, the higher the acceleration ratio. As the number of PCs increases, the relative acceleration ratio and parallel efficiency of the MPI parallel cloud-derived wind algorithm are shown in Figs. 11 and 12.

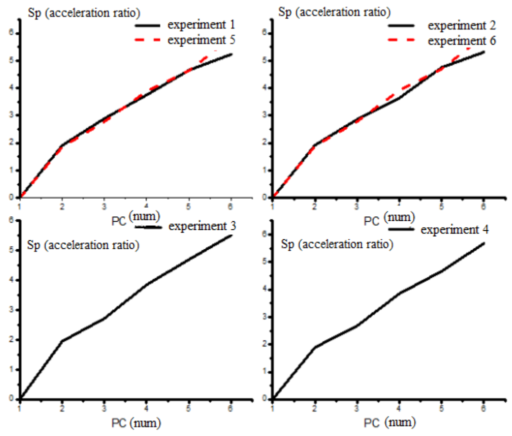

Figure 9: Acceleration ratio increasing with the number of PCs

We mainly use a multi-computer parallel method. As mentioned above, in addition to multi-computers, CPU multi-core and GPU parallel methods are also optional methods. The authors have achieved performance improvements of up to tens of times, with CPU multi-core and GPU parallel methods [Wang, He, Ouyang et al. (2016); He, Bai, Ouyang et al. (2019)]. We will not repeat them again.

6 Conclusion

In this paper, we discuss the parallel computing problem of cloud-derived wind inversion. A parallel algorithm based on computer cluster is designed and implemented, according to the characteristics of cloud-derived wind inversion. The analysis of the calculation results shows that assigning reasonable parallel granularity can guarantee the correctness of the calculation results and compared with the serial algorithm, it effectively improve the calculation efficiency. It also provides an efficient calculation for a short-interval, high-density and large-scale wind vector.

Recently the application of cluster computing is becoming more and more extensive, and the use of MPI for designing algorithm will be the direction to further improvement of the efficiency of cloud-derived wind inversion. In addition to the extensive 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, other tracer cloud tracking algorithms and the optimization of the tracking efficiency are considered.

Funding Statement: This work was supported in part by the National Natural Science Foundation of China (61872160, 51679105, 51809112); "Thirteenth Five Plan" Science and Technology Project of Education Department, Jilin Province (JJKH20200990KJ).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

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, pp. 1-9.

He, L. L.; Bai, H. T.; Ouyang, D. T.; Wang, C. S.; Wang, C. et al. (2019): Satellite cloud-derived wind inversion algorithm using GPU. *Computers, Materials & Continua*, vol. 60, no. 2, pp. 599-613.

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

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.

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.

Ren, Y. J.; Wang, J.; Feng, X. J.; Younn, G.; Kim, J. (2019): A hierarchical clustering based method to evaluate reuse of rare earth tailings under cloud computing environment. *Cluster Computing*, vol. 22, pp. 1805-1814.

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.

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, 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.

Zhang, W. (2015): Application research of computer cluster technology. *Practical Electronics*, no. 05, pp. 108-109.