Atmospheric and Climate Sciences, 2018, 8, 248-261
http://www.scirp.org/journal/acs
ISSN Online: 2160-0422
ISSN Print: 2160-0414

A Parallelization Research for FY Satellite Rainfall Estimate Day Knock off Product Algorithm

Weixia Lin, Xiangang Zhao, Cunqun Fan*, Manyun Lin, Lizi Xie

National Satellite Meteorological Center, Beijing, China
Email: fancq@cma.gov.cn

How to cite this paper: Lin, W.X., Zhao, X.G., Fan, C.Q., Lin, M.Y. and Xie, L.Z. (2018) A Parallelization Research for FY Satellite Rainfall Estimate Day Knock off Product Algorithm. *Atmospheric and Climate Sciences*, **8**, 248-261.
https://doi.org/10.4236/acs.2018.82016

Received: July 14, 2017
Accepted: April 27, 2018
Published: April 30, 2018

Abstract

With the development of satellite remote sensing technology, more and more requirements are put forward on the timeliness and stability of the satellite weather service system. The FY satellite rainfall estimate day knock off product algorithm runs longer, about 20 minutes, which affects the estimated rainfall product generated timeliness. Research and development of parallel optimization algorithms based on the needs of satellite meteorological services and their effectiveness in practical applications are necessary ways to enhance the high-performance and high-availability capabilities of satellite meteorological services. So aiming at this problem, we started the parallel algorithm research based on the analysis of precipitation estimation algorithm. Firstly, we explained the steps of precipitation estimated date knock off product algorithm; secondly, we analyzed the four main calculation module calculating the amount of algorithms; thirdly, multithreaded parallel algorithm and MPI parallelization was designed. Finally, the multithreaded parallel and MPI parallelization were realized. Experimental results show that the multithreaded parallel and MPI parallelization algorithm could greatly improve the overall degree of computational efficiency. And, MPI parallelization mode has a higher operating efficiency. The performance of parallel processing is closely related to the architecture of the computer. From the perspective of service scheduling and product algorithms, the MPI parallelization approach is adopted to achieve the purpose of improving service quality.

Keywords

Rainfall Estimate, Parallelization, Multithreading, MPI

1. Introduction

Fengyun meteorological satellites are mainly used in China's weather forecast, climate prediction and environmental monitoring and other aspects [1]. Meteorological satellite precipitation is estimated that the algorithm is a computationally intensive application, after the product is generated through the meteorological satellite product distribution system in a timely manner for all types of users to provide services. The current meteorological satellite precipitation algorithm in the IBM P780 high-end small machine running time is about 20 minutes, running longer [2]. Among them, the numerical prediction product data preprocessing, surface emissivity calculation, radiative transmission forward modeling and clear sky atmospheric precipitation inversion four calculation modules are computationally intensive and need to spend more computation time. This paper aims to study the parallelization method of data reconciliation algorithm for meteorological satellite precipitation, so as to improve the generation efficiency of precipitation products and improve the service level of precipitation products.

At present, there are many achievements in the parallelization of algorithms. In paper [3], it is pointed out that the parallel multithreaded architecture processor is composed of a plurality of logical processing machines. A large number of pipeline control parts are shared by all logical processors [4]. In each cycle, the processor fetches multiple instruction schedules from multiple threads. Lai Guoming *et al.* [5] detailed analysis of fast multi-pole algorithm FMM (Fast Multipole Method) the basic principles. He gives a description of the serial FMM algorithm. And the parallel analysis and processing of the FMM algorithm were carried out [6]. Su Shengqu and Liang Shizhen [7] pointed out that parallel computing as a modern computer is an important calculation method. This greatly optimizes the calculation process of the ant colony algorithm. Ant colony algorithm itself implies a certain degree of parallelism [8]. In essence, the ant colony algorithm is characterized by parallel cooperative optimization method, and the optimal solution is obtained by using parallel computation [9]. Guo Shen *et al.* [10] pointed out that the parallel compiler technology is the primary problem in the program can be found parallel. The parallelism of the program is characterized by the execution time of the program, the loop part of the program, the data dependency analysis and the execution time and the cycle times ratiop [11], and the parallel point mining in the program is carried out according to the above characteristics. Xu Jinxiu *et al.* [12] proceeded from the numerical simulation program of the Boltzmann model equation for solving the three-dimensional flow problem, and the MPI parallelization and high performance calculation and debugging of the numerical simulation program were carried out by studying the regional decomposition and parallel computing strategy [13]. A large-scale MPI parallel computing test was carried out to study the stability of the developed MPI parallelized region decomposition strategy and program optimization method [14].

The above-mentioned parallelization study has some reference significance. In order to solve the problem of long time to deal with meteorological satellite precipitation, we introduce multi-thread parallelization method and MPI (Message Passing Interface) parallelization method. And the parallel estimation and comparison of the precipitation estimation algorithm are based on these two methods in order to improve the calculation efficiency of the precipitation estimation algorithm.

2. Analysis of Estimation of Precipitation

2.1. Algorithm Steps

The calculation method of atmospheric precipitation inversion algorithm is as follows: 1) Read the required day VIRR L1 data including reflectance, radiation value, solar elevation angle, satellite observation angle information; 2) read into the numerical forecast product data, 3) read the MODIS land use type data (static), from the USGS, the temperature, the surface temperature, the surface pressure, For the five-minute product synthesis; 4) read the day of cloud detection data; 5) surface emissivity calculation processing package calculation according to VIRR5 minutes LIB data, cloud detection data, land surface cover classification data, early surface emissivity data calculation surface emission Rate; 6) numerical prediction data preprocessing package to interpolate the numerical prediction product data and convert the relative humidity of atmospheric observation into water vapor mixing ratio; 7) radiation transmission forward calculation According to the surface emissivity and the data after interpolation, the water vapor mixing ratio, Start the radiation transmission forward calculation, get the clear sky air precipitation inversion of the required mathematical parameters; 8) clear sky atmosphere Precipitation process packets start to clear sky atmospheric precipitation, the resulting data being submitted to public services like projection projected latitude and longitude.

2.2. Analysis of Algorithm Parallel Computation Module

VPWO algorithm mainly includes four processing modules, namely: 1) Numerical prediction of product data pretreatment; 2) Surface emissivity calculation; 3) Radiation transmission forward calculation; 4) Clear sky Atmospheric precipitation inversion. The following mainly gives the calculation of the four processing modules.

a) Preprocessing of numerical forecast product data

The numerical resolution of the obtained product data is lower than that of the remote sensing satellite observation data, and the spatial resolution is lower than that of the radiation mode. Therefore, it must be interpolated before it is calculated as the radiation transmission input data. Horizontal interpolation using binary in-range unequal interpolation algorithm, the numerical prediction field can be interpolated into orbital format. Vertical interpolation uses a one-whole interval interval interpolation algorithm, the original temperature and water vapor field can be interpolated to 43 layers of isobaric surface. High-level data are interpolated from American standard atmospheric profiles.

Horizontal interpolation: $640 \times 321 \rightarrow 1800 \times 2000$, inserted 14 + 17 layers; vertical interpolation: 1800×2000 points, 17 layers → 43 layers, 14 layers → 43 layers.

This section deals with the time in the business environment: Horizontal interpolation requires 40 threads. Processing time: 30 s. Vertical interpolation requires 20 threads. Processing time: 12 s. The vertical interpolation of the test on the simulation machine as follows Table 1:

b) Surface emissivity calculation

The surface emissivity calculation is statistically about 10 s, and the processing module is mainly time consuming to read data and interpolate.

c) Radiation transmission forward calculation

The input and output of the processing module are:

Input: including satellite azimuth, surface emissivity, temperature and humidity profile, surface temperature, surface pressure, ozone profile (after pretreatment);

Output: 43 layers of atmospheric atmospheric transmittance, in addition to the need to calculate the atmosphere of the K atmosphere relative to the atmosphere in order to calculate the radiation difference, the use of temperature difference, the process is more complex.

d) Clearance of airborne weather

According to the results of radiative forward calculation, the radiation difference and bright temperature difference of the two channels are calculated respectively. According to the difference of radiation difference, bright temperature difference, clear sky precipitation and surface emissivity, the atmospheric precipitation data can be calculated and output. The calculation is relatively small.

Radiation transmission forward calculation and clear sky atmospheric precipitation inversion two parts processing requires the use of 30 threads, processing time total 170 s.

2.3. Algorithm Process Parallel Relevance

Figure 2 shows the process: VPWObtgeneration (clear sky can be precipitation inversion process), VPWDayproduct (clear sky can be precipitation of water),

Table 1. Vertical interpolation test on the simulation machine.

number	Number of threads (s)	Time consuming (seconds)
1	20	10
2	40	6.9
3	50	6.5
4	60	4.8
5	70	7.3
6	100	8

the process of the parallel operation of the algorithm shown in Figure 1, the figure marked as follows: 1) Create (process of the creation of orders) 2) Process: VPWObtgeneration (VPWDayproduct), VCMObtproduct (VIRR segmentation cloud detection process); 3) Input interface: IFL_VIRR_LST_L3TENDAY (early VIRR4/5 channel surface emissivity product data), ISM_PGS_VIRR_L1 VIRR 5-minute segmented cloud detection data), VIRR_TPW_NWP (numerical prediction products, including temperature profile, humidity profile, surface temperature, surface pressure), ISM_PGS_LCD1KM (IGBP surface coverage classification data); 4) Output interface: IFL-VIRR_TPW_5MINR (VPWODibiaofashelv, VPWOSibzhiuubao, VPWOFushechuanshu, VPWOQingkongkejiangshui, and can be used to reduce the temperature of the water vapor Inversion processing package); 5) Public service: ZPRJ (Projection Transformation Utility Service Group), VPWOException (exception handling operation package).

3. Algorithm Parallelization

Multithreading parallelization and MPI parallelization are the most important means of algorithm parallelization. Multithreading is the process of dividing a program into several concurrent tasks. Each task works in parallel according to

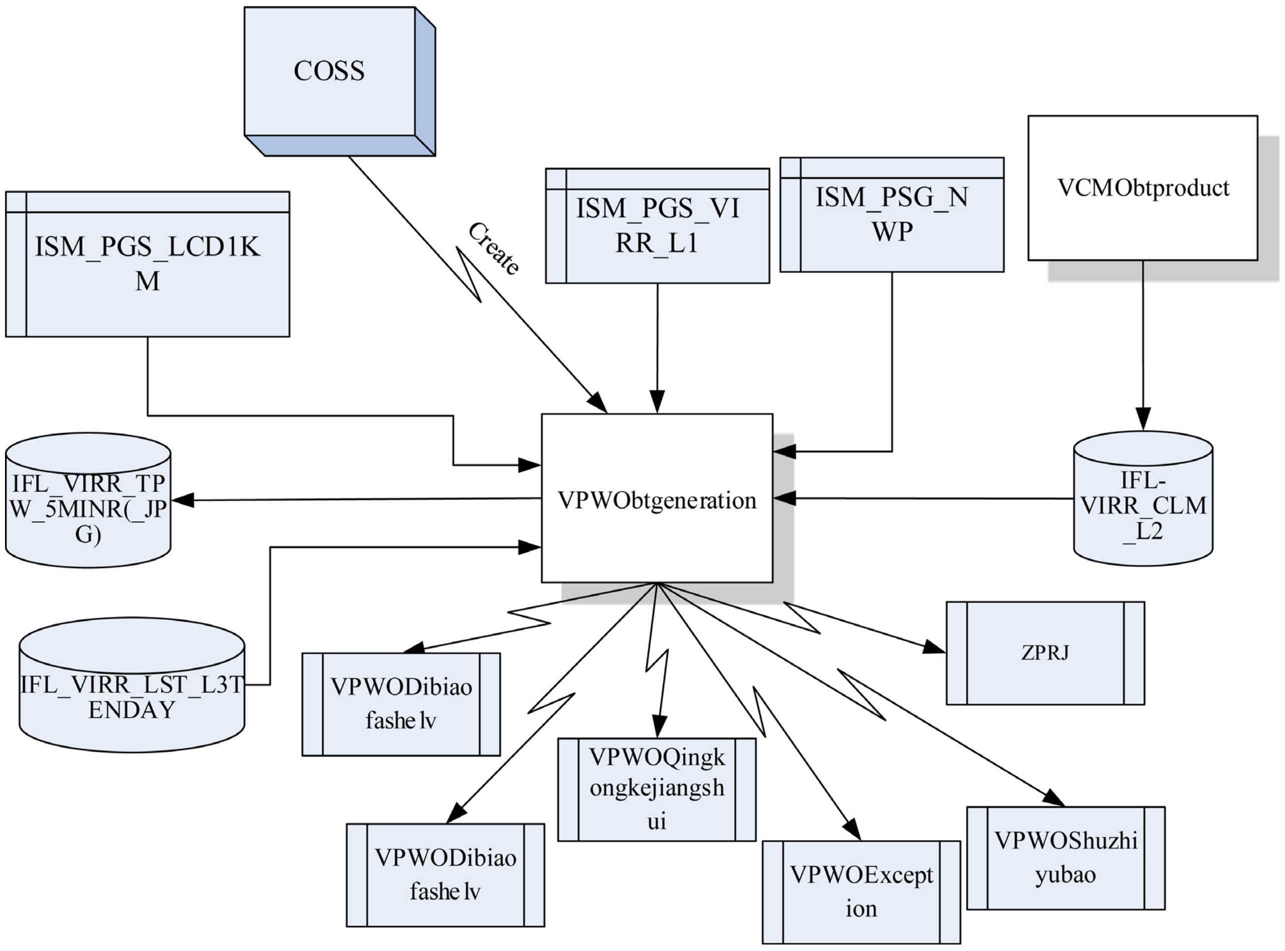

Figure 1. Process parallel operation associated graph.

different execution routes and performs its functions independently and does not interfere with each other. Multi-threading technology to achieve multi-processor hardware performance by accelerating program response, improving system throughput and resource utilization, and improving communication efficiency among multitasking. It can optimize the performance of the entire application system through multi-threading technology.

MPI is the messaging function library standard. It is a new library description. The library has a total of hundreds of function call interface. These functions can be called directly in Fortran and C languages. MPI as a messaging programming model. It implements data exchange between processors by explicitly sending and receiving messages. Each parallel process has its own independent address space, the process can not be directly accessible to each other, through the explicit message to achieve, the way for large-scale parallel processor (MPP) and cluster (Cluster). MPI is based on distributed memory, a distributed memory system consisting of a set of network-connected pairs of network connections. The memory associated with the kernel can only be accessed by the core. The process (or processor) has its own local memory. Different processes often need to interact with other processes to obtain and send data.

Based on the analysis of the precipitation estimation algorithm, this paper introduces the multi-thread parallel computing method [15] [16] and the MPI parallelization method [17] [18] to the parallel design of the algorithm. Therefore, two parallel architectures of precipitation estimation algorithm are proposed.

3.1. Multi-Threaded Parallelization

3.1.1. Multi-Threaded Parallelization Principle

Multi-thread parallelization mainly deals with multi-thread parallelization of four parallel modules in the precipitation estimation algorithm. The data is recorded by the main program VPWObtgeneration IFL_VIRR_LST_L3TENDAY, ISM_PGS_VIRR_L1, IFL_VIRR_CLM_L2, VIRR_TPW_NWP, ISM_PGS_LCD1KM. When the data is read, the sub-tasks are divided into several numerical predictive product data preprocessing, surface emissivity calculation, radiative transmission forward calculation and clear sky precipitation. The number of sub-tasks set according to the actual needs of the specific needs. In the experimental section we give a detailed analysis. Different sub-tasks assign a task thread to perform related task calculation processing. The calculated results are aggregated and merged at the output first. And finally output the data after the fusion. Figure 2 is a block diagram of the algorithm multi-threaded parallelization.

3.1.2. Pthreads Parallelization of the Algorithm

Based on the task parallel, the calculation part of the product inversion processing such as projection processing is repeated, and the calculation part is split into n tasks in parallel. Run the input command as./VPWD_F3C n, where n is the number of threads. Pshreads The pseudo-code of the parallelization

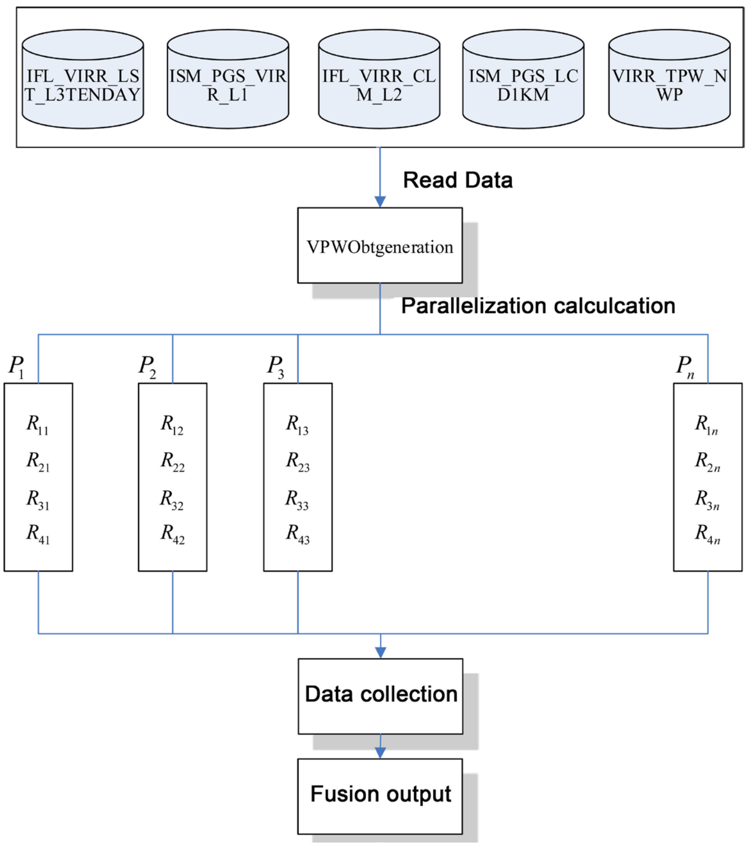
IFL_VIRR_LST_L3TENDAY
ISM_PGS_VIRR_L1
IFL_VIRR_CLM_L2
ISM_PGS_LCD1KM
VIRR_TPW_NWP
Read Data
VPWObtgeneration
Parallelization calculcation
P_1
P_2
P_3
P_n
R_{11} R_{21} R_{31} R_{41}
R_{12} R_{22} R_{32} R_{42}
R_{13} R_{23} R_{33} R_{43}
R_{1n} R_{2n} R_{3n} R_{4n}
Data collection
Fusion output

Figure 2. Schematic diagram of multi-threaded parallelization of precipitation estimation algorithm (**Figure 2** shows the thread, j is the thread identity, $j \in [1, n]$, R_{ij} is sub-task, where i is 4 parallelizable algorithm module identification, $i \in [1, 4]$, j is the subtask identification, $j \in [1, n]$.)

process is as follows:

```
    for(int i = 0; i < 576 ; i++)//The input data is complete source data
    {
    VpwdPrjProcess()
    {
        Thread_Count=strtol(argv[1],MULL,10);//Get the number of threads
         for（thread=0;  thread<Thread_Count;thread++）
        {
            //Create a thread, the inversion calculation into Thread_Count sub-tasks, by
Thread_Count threads in parallel
            pthread_create(VpwdProcess);
        }
        for（thread=0;  thread<Thread_Count;thread++）
        {
            //Blocking the call thread, until all the implementation of the end of the sub-thread,
the main thread to perform the remaining operations.
            pthread_join(thread_handles[thread]);
        }
        ……Other non-parallel operations
    }
    }
    WriteFile();//Generate precipitation products
    VpwdDelete();//Release memory
```

3.2. MPI Parallelization

3.2.1. MPI Parallelization Principle

MPI parallelism is achieved through data parallelism. The data files IFL_VIRR_LST_L3TENDAY, ISM_PGS_VIRR_L1, IFL_VIRR_CLM_L2, VIRR_TPW_NWP, ISM_PGS_LCD1KM and the projection lookup table are divided into n-sized modules. These modules are read by n compute nodes, respectively. And the numerical data of these data modules are preprocessed according to the computing power of each node, the calculation of surface emissivity, the calculation of radiative transmission and the calculation of the precipitation of airborne air. And the results of each node are aggregated to the master node, and the result is calculated by the fusion of the master node. Figure 3 is a block diagram of the algorithm MPI parallelization.

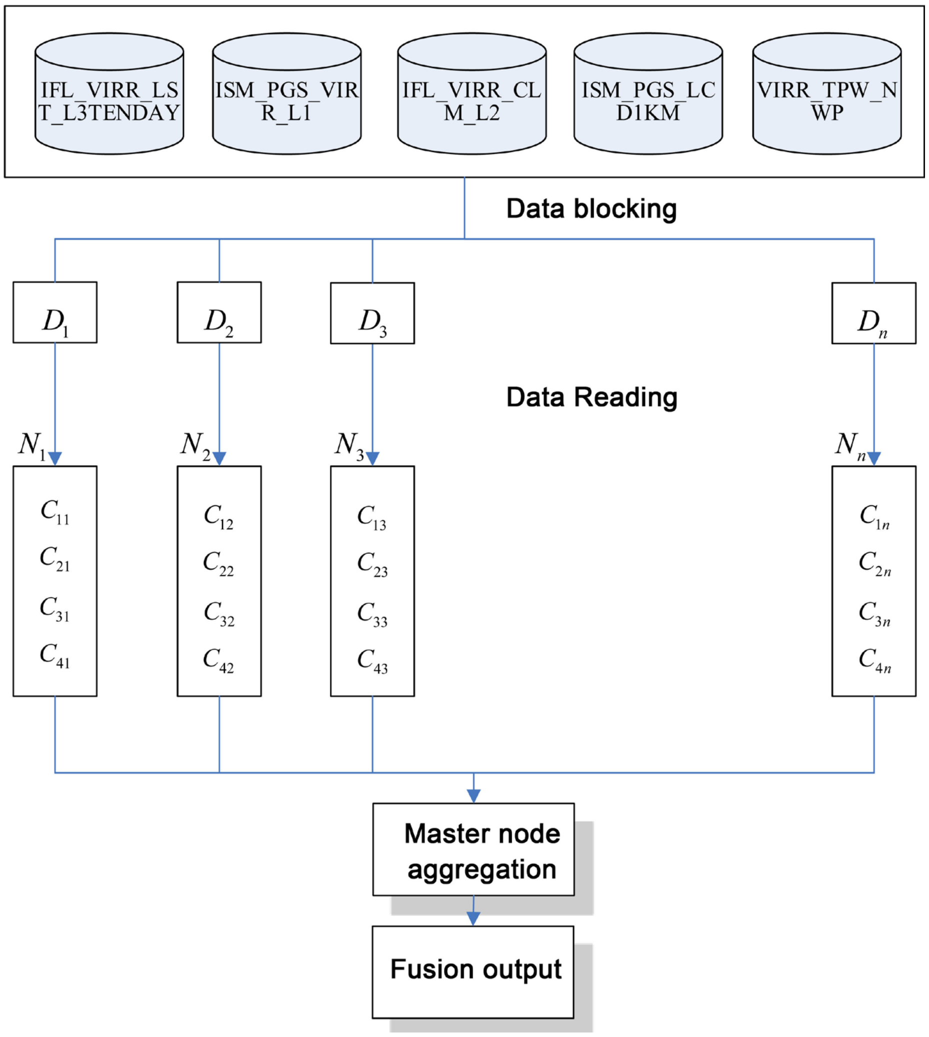

Figure 3. Precipitation estimation algorithm MPI parallelization block diagram (Figure 3 shows the data block, j is the data module identification, $j \in [1, n]$, N_j said the calculation node, j is the calculation node identification, $j \in [1, n]$, C_{ij} is the data module on the node N_j parallelization calculation, where i is the 4 parallelizable algorithm module identification, $i \in [1, 4]$).

3.2.1. MPI Parallelization of the Algorithm

Based on the data parallel, the input data track inversion file and the projection lookup table are divided into n equal sizes. And sent to the n nodes to perform the same projection processing and related computing operations. Parallel program using master-slave mode, the process number myid = 0 as the main process of the process. It receives the local processing result sent by MPI_Send by other processes via MPI_Recv call. Finally, the main process merges the local results to produce precipitation products. The operation configuration operation is as follows:

1) Edit the file ~/mpd.hosts. Fill in all machine names that allow access to this machine for parallel computing.

2) Generate the file ~/.mpd.conf,.mpd.conf file content: secretword = 123456, of which 123456 to identify the password.

3) generate ~/machinefile.

4) Start the mpd daemon.

5) Check whether each node is ready to run the mpi job.

6) Run the mpi parallel program.

The pseudo-code of the entire MPI parallelization process is as follows:

```
//MPIInitialize, the program starts in parallel
MPI_Init(&argc,&argv);    //MPI is initialized and the program starts in parallel
MPI_Comm_size(MPI_COMM_WORLD,&group_size);//Get the current process ID
MPI_Comm_rank(MPI_COMM_WORLD,&my_rank);//Gets the total number of running processes
//Each process enters data for the 1 / group_size block of source data
for(int i = 0; i < 576 /group_size; i++)
{
    //Inverse function of the product, 5 minutes of data read and projection, etc., to get local results
    VpwdPrjProcess();
}
if(my_rank!=0)//Non-master process
{
  MPI_Send(tempResult); //The non-master process sends the local projection calculation processing result to the main process    }
else //Main process
{
    MPI_Recv( tempResult);//The main process receives the local processing results from each child process
  WriteFile();//The main process generates precipitation products
  VpwdDelete();//Release memory
}
MPI_Finalize();// MPI parallelization ends
```

4. Experimental Results and Analysis

4.1. Experimental Configuration

The parallel precipitation estimation algorithm runs on the IBM P780 high-end machine. Its configuration is: CPU POWER7; memory 100G. In this experiment, HP server is used as the computing node. Four HP BL680G7 high-performance blade servers (housed in a C7000 10U blade chassis); two HP SL390s G7 (four in the C7000 10U blade chassis); four HP BL680G7 high-performance blade servers High-density servers (placed in a set of HP SL6500 4U chassis); landing and management nodes for the four HP DL380 G7 2U rack server, its configuration: CPU x86_64; memory 110G.

In this experiment, the input data of the precipitation day is 576 (day and night data each 288) block 5 minutes inversion file. In the MPI parallelization experiment, 576 files were divided. And send it to each node in parallel to perform the calculation. All the calculations in this experiment are based on hp-compute-27, and the results of other nodes are calculated.

4.2. Parallelization Comparison

The multi-thread parallelization of the precipitation estimation algorithm is related to the MPI parallelization function and the parallel mode as shown in Table 2.

4.3. Experimental Results and Analysis

In this paper, the parallelization algorithm based on multi-thread and the parallelization algorithm based on MPI are proposed for the parallelization of FY-3 satellite precipitation estimation. In order to verify the validity of the parallelization algorithm and compare it on the basis of this, we have realized the two parallelization methods respectively, and then the experimental results and the correlation analysis are given.

Figure 4 shows the results of the 10-day run of the precipitation estimation algorithm without parallelization. Running for about 15 minutes. Figure 5 shows the run-time results for multi-threaded parallelization algorithms with different numbers of threads. It can be seen that when the number of threads is gradually increased, the overall run time of the algorithm is gradually reduced. When the number of threads increased to 8, the increase in the number of threads does not make the overall run time has a more significant decline. The multi-thread parallelization method can stabilize the computation time of the whole algorithm in 180 s or so. Compared with the original algorithm for about 15 minutes of computing time cost, multi-threaded parallelization algorithm greatly improves the overall computational efficiency of the algorithm.

Table 2. Parallelization comparison.

Comparative	MPI	Pthread
Storage type	Distributed memory	Shared memory
Parallel type	Data in parallel	Task in parallel
Run the node	Multi - node cluster	Single node
Data sharing	Messaging	Direct delivery pointer
Program level	Process level	Thread level

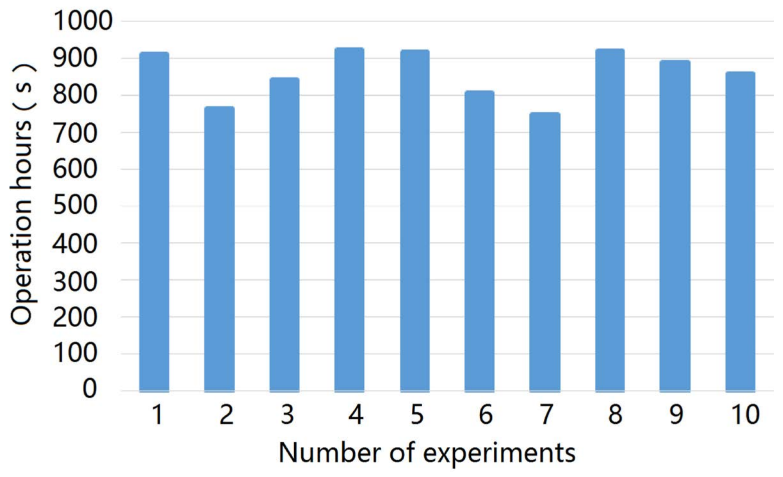

Figure 4. Precipitation estimation algorithm operation results.

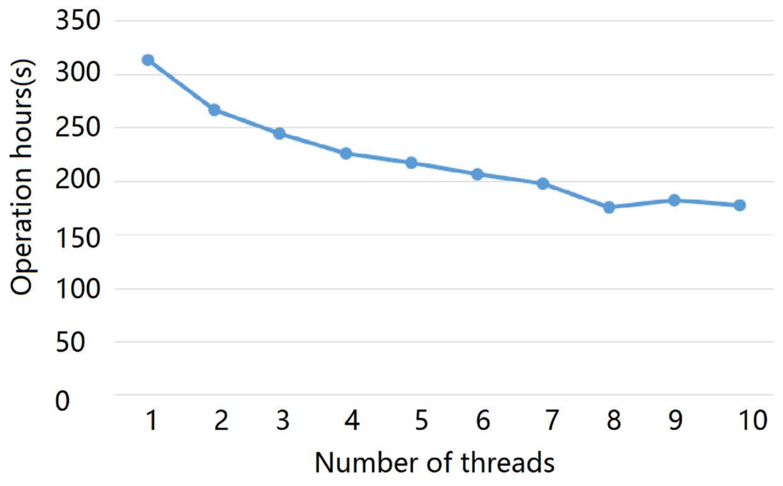

Figure 5. Multi-threaded parallel operation results.

Figure 6 shows the run-time results of the MPI parallelization algorithm at different processing nodes. It can be seen from the figure that when the computational node is changed from 1 to 2, the computation time of the algorithm is obviously reduced. When the number of nodes is increased step by step, The time cost is gradually reduced, when the number of processing nodes is 7, with the increase in the number of processing nodes, the overall operation time of the algorithm does not appear more obvious changes. In other words, the overall operation time of the algorithm is stable at around 70 s. The MPI parallelization method can further reduce the overall run time of the algorithm on the basis of the multi-thread parallelization method, compared with about 180 seconds when the algorithm is running for about 15 minutes and the multi-thread parallelization method. In order to further improve the parallelization of MPI and multi-thread parallelization of the algorithm to improve the effect of **Figure 7** shows the same node and thread in the case of two parallelization method of operating efficiency comparison.

As can be seen in **Figure 7**, multi-threaded parallelization and MPI paralleli

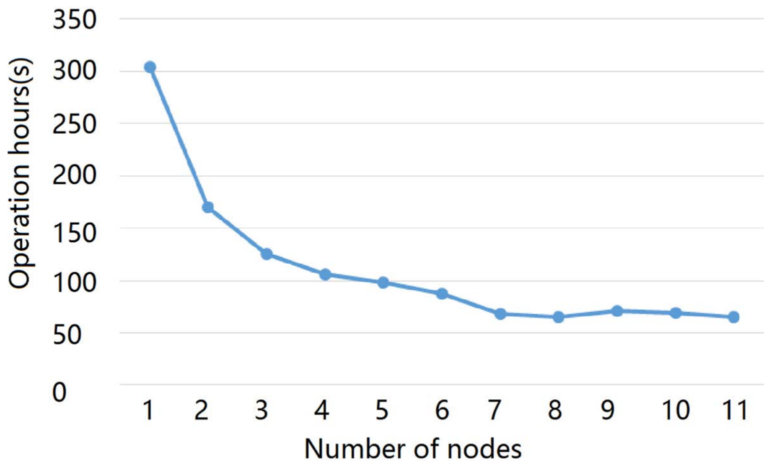

Figure 6. MPI parallelization results.

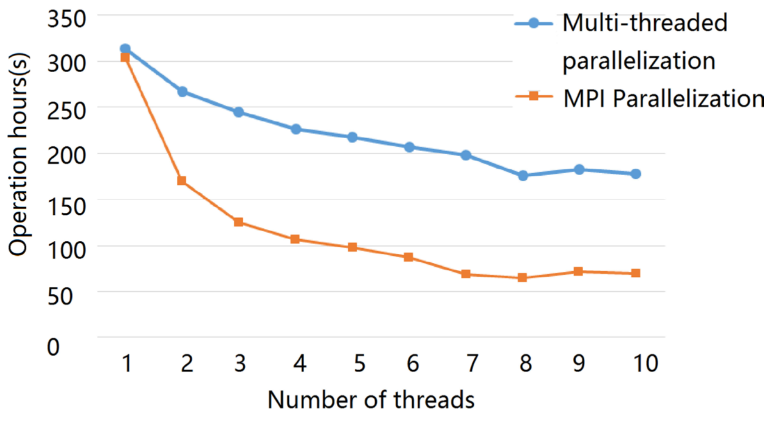

Figure 7. Parallelization run comparison.

zation algorithm in the same thread and node case, MPI parallel operation efficiency has a greater advantage.

In summary, the proposed multi-thread parallelization and MPI-based parallelization method can greatly improve the operational efficiency of the FY-3 satellite precipitation estimation algorithm. And MPI parallelization method has higher algorithm operation efficiency.

5. Conclusions

In this paper, we propose an algorithm based on parallelization method for the long time to deal with the estimation of FY-3 satellite precipitation. In this paper, the algorithm of estimating the daily estimation of precipitation is expounded and the four main calculation modules of the algorithm are analyzed. Finally, the algorithm is a multi-threaded parallelization and MPI parallelization design. Finally, the multi-MPI parallelization algorithm is implemented. And the algorithm is validated on the basis of this experiment. The processing time of multi-thread parallelization and MPI parallelization is obtained respectively. Experimental results show, the two parallel methods greatly improve the processing efficiency of the original algorithm. The experimental conclusion is as follows:

1) The processing efficiency of MPI parallelization method is Higher than the multi-threaded parallelization method. The parallelization research work is of great significance for the parallelization of other meteorological satellite data processing [13].

2) Parallel processing performance and computer architecture are closely related, not only depends on the CPU, but also with the system architecture, instruction structure, memory access speed and other factors [19].

3) Faced with large-scale mass data and high-dimensional data types, the traditional computing model has been difficult to provide the required processing power. The emergence of MPI parallel computing platform provides a new way for data processing [20].

Acknowledgements

The work presented in this study is supported by National High-tech R&D Program (2011AA12A104).

References

[1] Xu, J.M., Yang, J., Zhang, Z.X., *et al.* (2010) The Development and Application of Meteorological Satellites in China. *Meteorology*, **36**, 94-100.

[2] Celeux, G., Forbes, F. and Peyrard, N. (2001) Em Procedures Using Mean Field-Like Approximations for Markov Model-Based Image Segmentation. *Pattern Recognition,* **36**, 131-144. https://doi.org/10.1016/S0031-3203(02)00027-4

[3] Zhao, Q.M. (2005) Analysis of Parallel Multi-Thread Processor Architecture. *Microelectronics and Computer*, **5**, 185-187.

[4] Fernández-Pascual, R., Ros, A. and Acacio, M.E. (2016) Are Distributed Sharing Codes a Solution to the Scalability Problem of Coherence Directories in Many Cores? An Evaluation Study. *Journal of Supercomputing,* **72**, 1-27. https://doi.org/10.1007/s11227-015-1596-4

[5] Lai, G.M., Yang, S.Y. and Yuan, D.H. (2007) Parallelization of FMM Algorithm. *Journal of Computer Applications and Software*, **24**, 176-178.

[6] Kang, H.C., Kim, S.S., and Lee, C.H. (2015) Parallel Processing with the Subsystem Synthesis Method for Efficient Vehicle Analysis. *Journal of Mechanical Science & Technology,* **29**, 2663-2669. https://doi.org/10.1007/s12206-015-0512-4

[7] Su, S.Q. and Liang, S.Z. (2009) Parallelism of Ant Colony Algorithm. *Computer & Modern*, **10**, 18-20.

[8] Cecilia, J.M., Llanes, A., Abellán, J.L., Gómez-Luna, J., Chang, L.W., and Hwu, W.M.W. (2018) High-Throughput Ant Colony Optimization on Graphics Processing Units. *Journal of Parallel & Distributed Computing,* **113**, 261-274. https://doi.org/10.1016/j.jpdc.2017.12.002

[9] Kang, Y. and Tang, D. (2016) An Optimization Method for Meta-Functional Chain Design Solution Based on Computational Matrix and Ant Colony Algorithm. *Journal of Mechanical Engineering,* **52**, 25. https://doi.org/10.3901/JME.2016.23.025

[10] Guo, S., Li, P.F. and Zhu, Q.M. (2011) A Feature-Based Parallel Point Discovery Method. *Journal of Computer Applications and Software*, **28**, 24-26.

[11] Mitra, S. (2015) Method and System for Analyzing an Extent of Speedup Achievable

for an Application in a Heterogeneous System. *Human Reproduction*, **19**, 2738-2741.

[12] Xu, J.-X., Li, Z.-H. and Yin, W.-W. (2012) Application of MPI Parallel Debugging and Optimization Strategy in Numerical Simulation of Three-Dimensional Gas Flow Theory. *Computer Science*, **39**, 300-303.

[13] Afzal, A., Ansari, Z., Faizabadi, A.R. and Ramis, M.K. (2017) Parallelization Strategies for Computational Fluid Dynamics Software: State of the Art Review. *Archives of Computational Methods in Engineering*, **24**, 337-363. https://doi.org/10.1007/s11831-016-9165-4

[14] Volzer, T. and Eberhard, P. (2016) Model Order Reduction of Large-Scale Finite Element Systems in an mpi Parallelized Environment for Usage in Multibody Simulation. *Archive of Mechanical Engineering*, **63**, 475-494. https://doi.org/10.1515/meceng-2016-0027

[15] Xu, J.L., Jiang, L.H., Dong, W.Y., *et al*. (2011) Study on Multi-Thread Parallel Optimization of Dynamic Binary Translation. *Computer Engineering and Design*, **32**, 2370-2372.

[16] Mahafzah, B.A. (2013) Performance Assessment of Multithreaded Quicksort Algorithm on Simultaneous Multithreaded Architecture. *The Journal of Supercomputing*, **66**, 339-363. https://doi.org/10.1007/s11227-013-0910-2

[17] Dobosz, R., Hurley, R. and Mcconnell, S. (2015) A Hybrid Open MP-MPI Parallelization of Structure Software. *International Journal of Computer Applications*, **118**, 1-9. https://doi.org/10.5120/20786-3434

[18] Chenm, G., Sunm, G. and Xum, Y. (2008) Parallel Algorithm Research Method. *Journal of Computers*, **31**, 1493-1502.

[19] Nishiura, D., Furuichi, M. and Sakaguchi, H. (2015) Computational Performance of a Smoothed Particle Hydrodynamics Simulation for Shared-Memory Parallel Computing. *Computer Physics Communications*, **194**, 18-32. https://doi.org/10.1016/j.cpc.2015.04.006

[20] Minervini, M., Rusu, C., Damiano, M., Tucci, V., Bifone, A., and Gozzi, A., *et al*. (2015) Large-Scale Analysis of Neuroimaging Data on Commercial Clouds with Content-Aware Resource Allocation Strategies. *International Journal of High Performance Computing Applications*, **29**, 473-488. https://doi.org/10.1177/1094342013519483