Parallel Implementation and Optimization of a Hybrid Data Assimilation Algorithm

Jingmeifang Li and Weifei Wu(✉)

College of Computer Science and Technology,
Harbin Engineering University, Harbin 150001, China
{lijingmei, wuweifei}@hrbeu.edu.cn

Abstract. Data assimilation plays a very important role in numerical weather forecasting, and data assimilation algorithms are the core of data assimilation. The objective function of common data assimilation algorithms currently has a large amount of calculation, which takes more time to solve, thereby causing the time cost of the assimilation process to affect the timeliness of the numerical weather forecast. Aiming at an excellent hybrid data assimilation algorithm-dimension reduction projection four-dimensional variational algorithm that has appeared in recent years, the paper uses the MPI parallel programming model for parallel implementation and optimization of the algorithm, and effectively solves the problem of large computational complexity of the objective function. This effectively not only reduces the solution time of the algorithm's objective function, but also ensures the effect of assimilation. Experiments show that the speedup of the paralleled and optimized algorithm is about 17, 26, and 32 on 32, 64, and 128 processors, and the average speedup is about 26.

Keywords: Parallel · MPI · Data assimilation · Optimization

1 Introduction

Numerical weather forecasting is one of the important applications of high performance computing. It is a technique that solves a set of partial differential equations that describe the physical evolution of the atmosphere under the initial conditions that meet the conditions, so as to achieve a forecast for weather phenomena in the future [1]. As a forecasting problem, the quality of the initial value largely determines the accuracy of the numerical weather prediction. In numerical weather forecast, data assimilation technology is a technology that provides initial values for numerical weather prediction. It integrates observation information from various sources into the initial values of numerical weather prediction models. Based on strict mathematical theory, an optimal solution of the model is found between the model solution and the actual observation. This optimal solution can be used as the initial value of the numerical weather forecasting model which can be continuously recycled so that the result of the model solution continuously converges to the actual atmospheric state value.

The data assimilation algorithm is the core of data assimilation technology is. The objective function of the algorithm is usually a high-dimensional linear equation and calculation of the linear equations will be very large, so the solution of the objective

S. Liu and G. Yang (Eds.): ADHIP 2018, LNICST 279, pp. 306–314, 2019.
https://doi.org/10.1007/978-3-030-19086-6_34

function requires high-performance computer and parallel computing technology. The common data assimilation algorithms include variational algorithms (3Dvar and 4DVar), set Kalman filter algorithm (EnKF), and hybrid ensemble variational data assimilation algorithms based on the them [2, 3]. The hybrid ensemble variational data assimilation algorithm has advantages of variational algorithm and EnKF. Nowadays, many hybrid assimilation methods have emerged. The hybrid idea has become the trend of the data assimilation method. The research work of the world's top forecast centers is ongoing in this area. Among them, Dimension-Reduced Projection 4DVar (DRP4DVar) is an excellent hybrid assimilation algorithm. Compared with the commonly used algorithms, the DRP4DVar has the advantages of easy parallel implementation and small storage space. Therefore, the efficient implementation of the algorithm has gradually been concerned by researchers.

In this paper, through the analysis of DRP4DVar, the algorithm is paralleled and optimized by using MPI on the parallel computing platform, and an efficient implementation method of the object function standard solution of DRP4DVar algorithm is obtained. The structure of this paper is as follows: The 2 Section analyzes the serial implementation of DRP4DVar algorithm in detail, and then carries out parallel and optimization. The experiments are designed and the experimental results are analyzed in 3 Section. The 4 Section summaries the all work of the paper.

2 Analysis, Parallel Implementation and Optimization of DRP4DVar

2.1 Analysis and Implementation of DRP4DVar

Dimension-Reduced Projection 4Dvar (DRP4Dvar) is derived from the EnKF algorithm and the 4DVar algorithm. The principle of the algorithm is to use a set of vectors generated by a set of perturbation sets, and to perform the projection operation on the basis vectors of the analysis increments of the model. After the projection is completed, the solution of the objective function is transformed into the solution of the coefficients of the set of basis vectors, which completes the projection of the model space to the base vector [4, 5]. Because the dimension of the sample space is small, it will generally not exceed 10^2, so the objective function dimension of the algorithm will not exceed 10^2. Therefore, theoretically the solution size of the algorithm will not exceed 10^2, and because of the inherent parallelism of the ensemble, the algorithm has good parallel efficiency. The improved objective function of DRP4DVar is formula (1).

$$\begin{cases} J(w) = \dfrac{1}{2}\{w^T w + \dfrac{1}{2}[Y' - r_b w]^T O^{-1}[Y' - r_b w]\} \\ Y^o = (y_0^o, y_1^o, \ldots, y_N^o)^T \\ F(x) = (H_0 \circ M_{t0->t1}(x), H_1 \circ M_{t0->t2}(x), \ldots, H_n \circ M_{t0->tn}(x))^T \end{cases} \tag{1}$$

In formula (1), the pattern space contains n samples, and n samples are represented as vector $x = [x_1, x_2, \ldots\ldots, x_n]$, averaging all values of n samples to get x_b as

background field, also called initial value. Y^o is an observation vector containing an assimilation time, and also known as an observation space vector. $F(x)$ is the vector of $H_0 \circ M_{t0->t1}(\boldsymbol{x})$ at each moment, also known as the analog observation vectors. H_i is the observation operator at time t_i, $M_{t0->ti}$ is the model forecast operator from t_0 forecast to t_i.O is the observation error covariance matrix. $w = [w_1, w_2, w_3, \ldots, w_n]^T$ is an n-dimensional coefficient vector. $Y' = Y^o - F(x_b)$.

The sample ensemble vectors minus the background field vectors, which yield a perturbed ensemble vectors in the pattern spaces:

$$b = [x', x', \ldots\ldots, x'] = [x_1 - x_b, x_2 - x_b, \ldots\ldots, x_n - x_b] \tag{2}$$

The perturbation vector δx can be expressed as a linear combination of column vectors of b:

$$\delta x = x - x_b = w_1 x_1' + w_2 x_2' + \ldots + w_n x_n' = bw \tag{3}$$

The perturbation ensemble of the observation space is formula (4):

$$r_b = [F(x_1) - F(x_b), F(x_2) - F(x_b), \ldots, F(x_n) - F(x_b)] \tag{4}$$

Gradient of the cost function in formula (1):

$$\Delta J(w) = w + r_b^T O^{-1}(r_b w - Y') \tag{5}$$

Let formula (5) equal to 0:

$$(I + r_b^T O^{-1} r_b)w = r_b^T Y' \tag{6}$$

Solving the formula (6) can get the value of w, substituting the value of w into the formula (3), and adding the δx to the background field x_b to get the optimal analysis value:

$$x_a = x_b + \delta x \tag{7}$$

The dimension of Eq. (6) is n, where n is the number of samples and does not exceed 10^2 at most. Therefore, the formula (6) has a small dimension and the solution is easy. However, similar to EnKF, the lack of sample will also result in insufficient estimation of the background error covariance matrix, which will result in inaccurate results. In order to alleviate the impact of insufficient samples, localization technology is also needed [6, 7]. The coefficient matrix obtained by localization of the observation space is py, the dimension is $my \times mb$, mb is the localized modal number, my is the number of observation points. The coefficient matrix obtained by localization of the model space is px, and the dimension is $mx \times mb$, mb is also the localized modal number, mx is the number of grid points in the horizontal direction.

py and r_b do Shure product to get r_b', px and b do Shure product to get b'. Using r_b' instead of r_b, b' instead of b, in formula (3) and (6), we can get the following formulas:

$$\delta x = b'w \tag{8}$$

$$(I + (r_b')^T O^{-1}(r_b'))w = r_b'^T Y' \tag{9}$$

$$(I + (\frac{r_b}{\sqrt{o}})'^T(\frac{r_b}{\sqrt{o}})')w = (\frac{r_b}{\sqrt{o}})'^T Y' \tag{10}$$

Solve Eq. (10) using the modified conjugate gradient algorithm and solve it strictly according to the mathematical formula. The detailed solution processes are as follows:

1. Given an initial value w_0, $d_0 = r^0$, computing $r_0 = b - Aw_0$.
2. Compute $\alpha^k = <r^k, d^k> / <d^k, Ad^k>$.
3. Compute $w_{k+1} = w_k + \alpha^k d_k$.
4. Compute $r^{k+1} = b - Aw_{k+1}$.
5. Judge whether w_{k+1} meets the requirements, and if it does, stop the calculation, otherwise, continue the subsequent steps.
6. Compute $\beta^k = -<r^{k+1}, Ad^k> / <d^k, Ad^k>$
7. Compute $d^{k+1} = r^{k+1} + \beta^k d_k$, return step 1 and continue.

However, localization must be added to reduce the influence of the false correlation of the estimated background error covariance matrix on the analysis results due to the lack of sample size. After localization, the dimensionality of the cost function formula (10) is on the order of 10^{10}. At this point, matrix A requires approximately 190 GB of storage space, which has exceeded the memory size of the most processors. Therefore, formula (10) cannot be calculated directly.

In order to solve the problem of matrix A storage difficulty, in view of the relatively powerful computing power of the processor, computing is cheaper than storage, so a serial implementation strategy is designed by exchanging computing for storage. The core idea of this strategy is: When the matrix A is used in the algorithm, the intermediate results that interact with A are processed first. The matrix A is not stored directly, that is, only the form of the matrix A is retained, and the specific matrix A is not stored. The strategy uses the processor's powerful computing power in exchange for storage of matrix A.

The core functions of formula (10) include: the grd function for calculating the residual, the ax function for calculating Ad^k, the function ddot3 for the vector dot multiplication operation, and the pybo operation function for calculating the right end vector. The parallel implementation of the algorithm is: the above conjugate gradient algorithm is solved on the root processor, and then the result is sent to all processors by utilizing the MPI communication function to perform formula (7) and (8) calculation.

The storage space required for the above objective function implementation of the DRP4DVar algorithm is greatly reduced, but it is less optimistic in the consumption time of the calculation, especially when running a large-scale case, the consumption time is longer, the real-time measurement under the high-resolution case about two

hours or more. In order to shorten the calculation time of the DRP4DVar algorithm, optimization work is also required.

2.2 Optimization of DRP4DVar Algorithm

Based on the DRP4DVar algorithm in Sect. 2.1, in order to improve the computational efficiency of the algorithm and shorten the running time, the algorithm is optimized. After many tests, the function pybo, grd, and ax calculations take the most time, so the optimization work is done for them.

After analyzing the three functions, the loop operations are similar, and the number of loops is about $nn \times jpx \times jpy \times jpz \times myb$, where nn is the number of samples, and the maximum is no more than 10^2, jpx, jpy, and jpz are the number of localization coefficients in three directions of horizontal and vertical, respectively. The number of coefficients does not exceed 10^2. The parameter myb indicates that the number of observations is on the order of 10^5. Therefore, the loop magnitude is about 10^{11}. At present, the most processors have a frequency of 10^9, so each of these three operations requires at least 100 s. Therefore, the purpose of optimization is to reduce the size of the loop, which in turn will reduce the time-consumption.

The parallel implementation strategy in Sect. 2.2 is improved as follows:

(1) Filter the observation data according to the corresponding relationship with the background field and remove the abnormal observation points.
(2) In the root processor, observation data and background field data are divided in all processors according to geographic location information.
3) The root processor uses the MPI sending policy to distribute the divided data to all processors, so that each processor is allocated a reasonable amount of data, avoiding the load imbalance on some processors. The improvement strategy reduces the computational size on each processor, thereby reducing computational time-consuming and improving the overall computational efficiency.

Through optimization operations, the time consumed by the DRP4DVar algorithm is theoretically reduced by an order of magnitude compared to the pre-optimization. Therefore, when calculating large-scale calculations, the time-consumption can also be the ideal range.

3 Experimental Design and Analysis of Results

To evaluate the benefit of the optimized DRP4DVar, we design many experiments. The experimental platform is a Linux cluster, which includes one login node and 16 computing nodes. Each compute node contains 2 physical CPU cores, and each physical CPU has 6 physical cores. Therefore, the test platform can provide up to 192 cores of computing resources. The detailed parameters of the computing node are shown in Table 1.

Table 1. Node parameters

Name	Parameters
CPU	Intel(R) Xeon(R) X5650@ 2.67 GHz
RAM	46 GB
Operating System	Red Hat Enterprise Linux Server release 6.3 (Santiago)
System kernel	2.6.32–279.el6.x86_64
C compiler	icc version 15.0.2
Fortran compiler	ifort version 15.0.2
MPI version	Intel(R) MPI Library 5.0

3.1 Experimental Design

In order to test the effect of the optimized DRP4DVar algorithm, two kinds of examples are designed: low-resolution case and high-resolution case. High-resolution case contains more data, low-resolution case contains less data. The cases can test the calculation efficiency under different calculations.

Case 1 is a low-resolution case where the simulation area is 129 * 70 horizontal grid points and 50 vertical layers. Case 2 is a high-resolution case where the simulation area is a horizontal grid point is 409 * 369, 29 vertical layers. In Case 1, the number of observations used for assimilation is 15729 and 2225 observations are used in Case 2. The number of observations will affect the solution time of the core conjugate gradient algorithm in the algorithm implementation, and the resolution will affect the calculation time of the initial value increment of the model. Two different kinds of experiments were set up according to two cases: low-resolution experiment and high-resolution experiment. Low-resolution experiment uses the algorithm before and after optimization to simulate case 1 under three parallel degrees on 32, 64, and 128 processors. High-resolution experiment uses the number of processors in three parallel degrees 32, 64, and 128, simulating algorithm before and after optimization for case 2.

The following information can be obtained by analyzing the results of two groups of experiments: Under low-resolution examples, the pre-optimization and post-optimization algorithms achieve time-contrast information under the same degree of parallelism. Similarly, under the high-resolution case. Under the same degree of parallelism, the time-consumption and results by the simulating case 1 and the case 2 are compared between the pre-optimization and post-optimization algorithms.

3.2 Results Analysis

For the two kinds of experimental conditions designed in Sect. 3.1, the experimental results were analyzed. The analysis includes the following two parts: the comparison of the results between before and after optimization, the comparison of consumption time between before and after optimization Therefore, this section is divided into correctness analysis and performance analysis.

(1) Correctness analysis

Figures 1 and 2 are incremental graphs of the four assimilation variables T, U, V and Q, respectively, in the serial implementation of the algorithm and the parallel implementation. It can be seen that the size of the increments of the four assimilation variables in the two results is exactly the same, and the form of the field of each variable is also the same. Therefore, the result of the algorithm optimization is exactly the same as the result before the optimization, so the result after the optimization is correct.

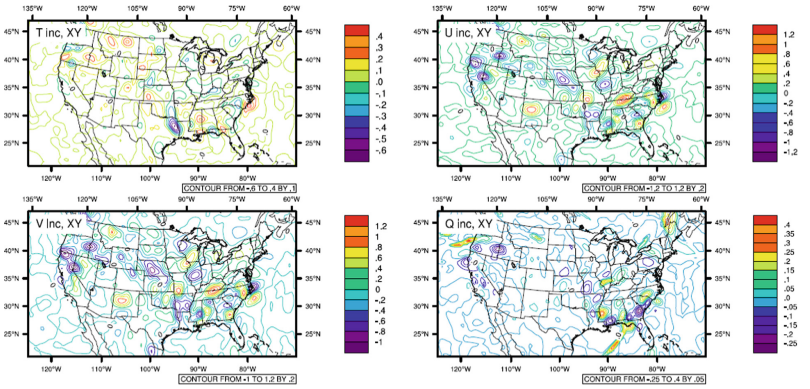

Fig. 1. Incremental graph of four assimilation variables by DRP4DVar serial implementation

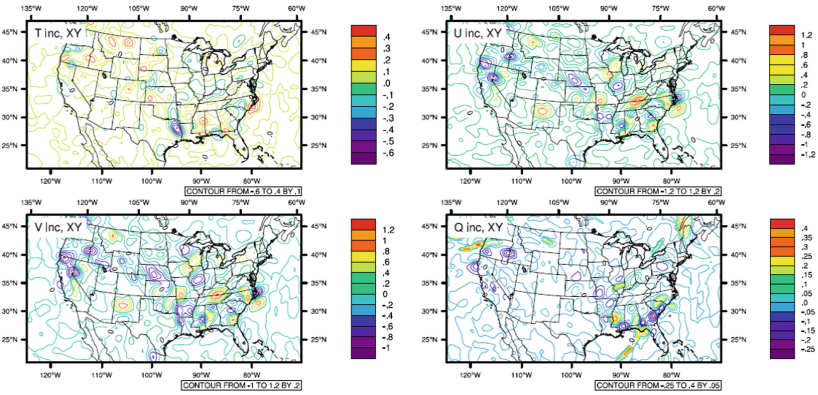

Fig. 2. Incremental graph of four assimilation variables by DRP4DVar parallel implementation

(2) Performance analysis

The first is a low-resolution case where 32, 64, and 128 processors are used to simulate case 1 respectively. Figure 3 is a comparison chart of the time consumed by running the calculation case 1 between pre-optimization and post-optimization of the algorithm.

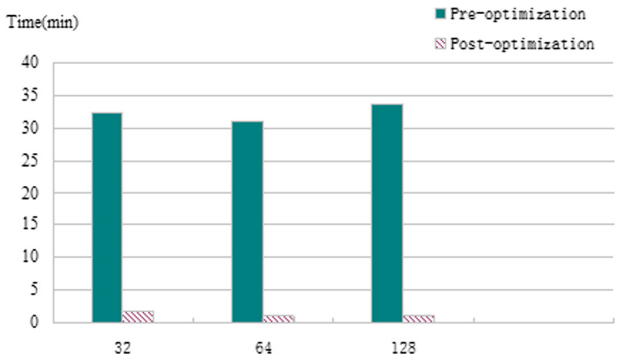

Fig. 3. Time-consumption of simulating case 1 between pre-optimization and post-optimization under three parallel degrees

From Fig. 3, it can be concluded that on the three processor numbers in case 1, the consumption time between pre-optimization and post-optimization has been significantly reduced.

On 32 processors, the time-consumption before optimization is about 18 times of post-optimization, on 64 processors the times is about 26 and on 128 processors the pre-optimization is about 33 times comparing with post-optimization.

Through analysis, it can be seen that the time of the optimized algorithm under the low-resolution case 1 is greatly reduced. Under the three types of processors, the optimized consumption time is about 1/26 of the time before the optimization, that is, the parallel speedup approximately reaches 26.

It can be concluded from Fig. 4 that in the case of the three processors, the consumption time before and after optimization is also significantly reduced. It can be seen from the analysis that the simulation time of the optimized algorithm in the high-resolution case 2 is reduced compared with the pre-optimization. Under the three processor numbers, the consumption time of post-optimization is about 1/4 of the pre-optimization. The speedup reaches about 4.

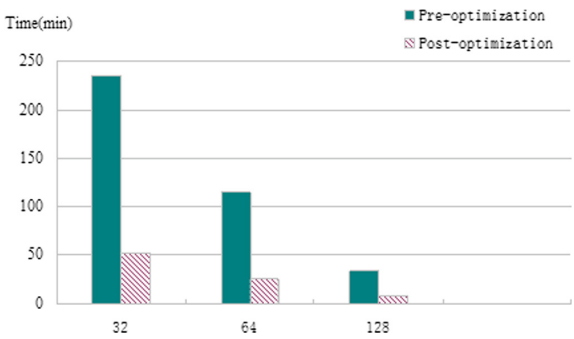

Fig. 4. Time-consumption of simulating case 2 between pre-optimization and post-optimization under three parallel degrees

4 Conclusions

In this paper, the DRP4DVar algorithm in the hybrid data assimilation algorithm is analyzed. Because of the serial implementation costing much time, the MPI parallel programming model is used for parallel implementation and optimization to obtain an efficient implementation of DRP4DVar. Then, the design test schemes verify the optimized algorithm implementation. The experimental results show that the implementation of DRP4DVar algorithm is accurate and the time performance of the algorithm is improved. In the low-resolution case, the parallel speedup is about 26, and the speedup is about 4 in the high-resolution case.

Acknowledge. This work was supported by the National Key Research and Development Plan of China under Grant No. 2016YFB0801004.

References

1. Miyoshi, T., Kondo, K., Terasaki, K.: Big ensemble data assimilation in numerical weather prediction. Computer **48**(11), 15–21 (2015)
2. Nerger, L., Hiller, W.: Software for ensemble-based data assimilation systems-Implementation strategies and scalability. Comput. Geosci. **55**(5), 110–118 (2013)
3. Ma, J., Qin, S.: The research status review of data assimilation algorithm. Adv. Earth Sci. **27**(7), 747–757 (2012)
4. Wang, B., Liu, J., Wang, S., Cheng, W., et al.: An economical approach to four-dimensional variational data assimilation. Adv. Atmos. Sci. **27**(4), 715–727 (2010)
5. Liu, J., Wang, B., Xiao, Q.: An evaluation study of the DRP-4-DVar approach with the Lorenz-96 model. Tellus Series A-Dyn. Meteorol. Oceanogr. **63**(2), 256–262 (2011)
6. Han, P., Shu, H., Xu, J.: A comparative study of background error covariance localization in EnKF data assimilation. Adv. Earth Sci. (2014)
7. Evensen, G.: The Ensemble Kalman Filter: theoretical formulation and practical implementation. Ocean Dyn. **53**, 343–367 (2003)