Journal of Science and Engineering Research
Vol. 1, No. 2, 2022, pp. 21-34
eISSN: 2957-7187

Particle Swarm Optimization Algorithm for Data Distribution Model Design

David Taniar

Monash University, Clayton, Australia
david.taniar@Monash.edu

DOI: http://dx.doi.org/10.56828/jser.2022.1.2.2
Article history: *Received (February 10, 2022); Review Result (March 23, 2022); Accepted (June 4, 2022)*

Abstract: This paper studies the data distribution problem of full comparative computing, proposes a data distribution model and related algorithms based on Particle Swarm Optimization (PSO) algorithm, and conducts experiments on the algorithm. The experimental results show that the data distribution scheme given by the data distribution model of the particle swarm optimization algorithm can realize the complete localization of the data files required by the task, and can reduce the use of storage space in the distributed system. In terms of load balancing, the task scheduling scheme given by this model can achieve load balancing among various nodes in a distributed system. In terms of computing time, the model can find data distribution schemes and task scheduling schemes at a faster rate and can better complete the data distribution of full comparison calculations in a distributed system. The data distribution model of the particle swarm optimization algorithm effectively solves the data distribution problem of large-scale full-comparison computing and will promote the research progress of bioinformatics, natural language processing, and other fields.

Keywords: Data distribution, Particle swarm optimization, Load balancing, Hadoop

1. Introduction

Full comparison calculation is a typical calculation mode [1], used to solve a type of calculation associated with pairwise data files. Full-comparative computing, as a special computing model, frequently appears in many disciplines, such as bioinformatics [2], biometrics [3], traditional machine learning field [4][5][6], natural language processing field [7], the field of traffic big data [8][9]. Scholars at home and abroad have been researching full-comparative computing, and full-comparative computing is one of the research hotspots. In foreign countries, some scholars have copied all the data required for the full comparison task on each computing node in the distributed cluster [10]. This distribution method is suitable for small data volumes and will cause serious network congestion and waste of storage space in the face of massive data. Someone used Hadoop Distributed File System (HDFS) to store the data needed to perform full comparison tasks [11]. HDFS uses a distributed copy storage solution, and this component uses a storage solution with 3 copies by default. Although this data storage method can save storage space, it cannot guarantee the complete localization of the data when performing comparison tasks. Chaudhary et al. built a heterogeneous computing platform when analyzing biological sequences. To achieve load balancing of the entire system, they allocate tasks according to the hardware configuration of the nodes. In terms of data distribution, they split the database and then distribute it to various nodes. Although a heterogeneous computing platform is used for calculation, it is still unavoidable to request data from other nodes in the cluster [12]. For the general full comparison data distribution scheme, some scholars have proposed the use of heuristic schemes for data distribution and task scheduling of full comparison calculations [13]. This method uses enumeration to obtain the optimal data distribution plan. When encountering large data sets, this method will become an NP-hard problem. Based on this research, the scholar put forward the idea of using a simulated annealing algorithm to construct a meta-heuristic data distribution algorithm, which can effectively reduce the use of storage space in distributed systems [14][15][16][17][18].

In China, some scholars have proposed the use of graph coverage to allocate data for full comparison tasks and proposed the DAABGC algorithm based on graph coverage theory [19]. The DAABGC algorithm can obtain a better data distribution scheme under the condition that the number of files is the same as the number of nodes, but it is not suitable for scenarios where the number of data files and the number of nodes are different. In the previous full comparative computing research, the branch and bound method were used to complete the data distribution of the full comparison calculation. Although this method can obtain the optimal data distribution plan, it needs to sacrifice a certain amount of solution time.

At present, there are still some problems in the data distribution research results of full comparative computing. In this paper, the theoretical analysis of the data distribution problem of full comparative calculation is further carried out. The problem is combined with the idea of Particle Swarm Optimization, and a data distribution model is proposed. Swarm optimization realizes load balancing data distribution algorithm and particle swarm optimization realizes optimal storage data distribution algorithm.

Finally, the feasibility of the model is verified based on MATLAB, and the related experiments are compared with the data distribution strategy of the Hadoop framework to verify that the algorithm is in the distributed system in terms of load balancing, storage space occupation, data localization, and algorithm operation speed.

2. Basic theoretical research

2.1. Full comparison calculation

In the data set of the full comparison calculation, the comparison calculation between the two data must only occur once. In a data set with m data files and a distributed cluster of n nodes, let the specific comparison algorithm be $C(i,j)$, where i and j are the two data files in the data set. The full comparison calculation can be formally described as formula (1). Among them, $M_{i,j}$ is the calculation result of the comparison operation $C(i,j)$.

$$M_{i,j} = \{\, C(i,j) \mid i < j, i = 1,2,\cdots,m-1, j = 2,3,\cdots,m \,\} \tag{1}$$

2.2. Data distribution model construction

The data distribution work in a homogeneous distributed system, assuming that the software and hardware configurations of each node in the distributed system are consistent. In the nucleic acid sequence alignment, the size of the data file is the same or approximately the same. Suppose there are m data files in the data set, and the size of each data file is s_i,

$i = 1,2,\cdots,m$. Then the formal description of the same or approximately the same size of the data file is as follows:

$$s_1 \cong s_2 \cong \cdots \cong s_m \tag{2}$$

The data distribution work of full comparison calculation should balance the load between nodes and reduce the overall storage space usage of the distributed system. At the same time, it is necessary to ensure that each comparison task can use data with localized attributes and to reduce the storage space on each computing node as much as possible. During the comparison operation of the two data files, the sum of the size of the two data files will be proportional to the calculation amount of the comparison operation. Let c_{ij} denote the calculation amount of the comparison task $C(i,j)$. Then c_{ij} has the following relationship with data file i and data file j:

$$s_i + s_j \propto c_{ij}, i < j, i = 1,2,\cdots,m-1, j = 2,3,\cdots,m \tag{3}$$

According to formula (2), it can be known that all the values of c_{ij} will be the same or approximately the same. Assuming that K comparison tasks are allocated to node p, the calculation amount of each task is c_{ij}^k, i and j are the numbers of the data files distributed to node p, and the task volume size p_c on node p is:

$$p_c = \sum_{k=1}^{K} c_{ij}^k, i,j \in p \tag{4}$$

Let t_{count} be the total number of tasks in the full comparison calculation. A comparison task is related to two different data files. When there are m data files in the data set, t_{count} is:

$$t_{count} = \binom{m}{2} = \frac{m(m-1)}{2} \tag{5}$$

Use x_{kp} to indicate whether the comparison task k is allocated to node p. From formula (1), it can be seen that x_{kp} is unique. When task allocation is performed, each comparison task can and must be allocated to a computing node. The value of x_{kp} is used 0 and 1 are used to indicate, 1 indicates that the comparison task k is allocated to the computing node p, and 0 indicates that no allocation is performed.

$$x_{kp} = 0or1, k = 1,2,\cdots,t_{count}, p = 1,2,\cdots,n \tag{6}$$

Let c_k represent the size of the k-th task, and then the task amount p_c on node p in formula (4) can be redefined as:

$$p_c = \sum_{k=1}^{K} (c_k x_{kp}), p = 1,2,\cdots,n \tag{7}$$

In a distributed system with n computing nodes, c_k is used to represent the size of task k, and the average calculation amount of nodes c_{avg} for full comparison calculation is:

$$c_{avg} = \frac{1}{n} \sum_{k=1}^{t_{comt}} c_k \tag{8}$$

From formula (7) and formula (8), the degree of deviation f_p of the calculation amount of node p relative to c_{avg} can be obtained.

$$f_p = |p_c - c_{avg}| \tag{9}$$

Then the load balancing of the distributed system can be described as;

$$min \sum_{p=1}^{n} f_p \tag{10}$$

Let w_q denote the total size of files distributed to node p. Suppose the number of files distributed to node p is $m'(m' \leqslant m)$, and the size of file i in w_p is r_i. Then there are:

$$w_p = \sum_{i=1}^{m'} r_i, p = 1,2,\cdots,n \tag{11}$$

Let $u_p(p = 1,2,\cdots,n)$ be the upper limit of the storage space of node p, and the total size of the files distributed to node p cannot exceed u_p.

$$w_p \leqslant u_p \tag{12}$$

Taking formula (10) as the solution goal, the load balancing solution model of the full comparison calculation in the distributed system is obtained as follows:

$$min \sum_{p=1}^{n} \left| \sum_{k=1}^{\frac{m(m-1)}{2}} c_k x_{kp} - \frac{1}{n} \sum_{k=1}^{\frac{m(m-1)}{2}} c_k \right|$$
$$s.t. \begin{cases} \sum_{p=1}^{n} x_{kp} = 1, k = 1,2,\cdots,\frac{m(m-1)}{2} \\ \sum_{i=1}^{m'} r_i \leqslant u_p, p = 1,2,\cdots,n, m' \leqslant m \\ x_{kp} = 0 \ or \ 1, k = 1,2,\cdots,\frac{m(m-1)}{2}, p = 1,2,\cdots,n \end{cases} \tag{13}$$

According to formula (13), the scheduling of the full comparison task in the distributed system will enable the load balance among the various computing nodes and can ensure that the data files required by each comparison task have data locality. On this basis, the model is optimized, looking forward to finding a data distribution plan that minimizes the storage space required in the distributed system.

There is a many-to-many relationship between data files and comparison tasks, that is, the same data file is related to multiple comparison tasks, and a comparison task requires two data files. Based on this many-to-many relationship, the task allocation is reorganized, the computing node that executes the comparison task k is adjusted, the data file distribution plan is modified, and a data distribution plan and a task scheduling plan are obtained.

The calculation amount of each node in the load balancing state can be obtained by formula (13), which is denoted as $g_i(i = 1,2,\cdots,n)$. There may be numerical differences between g_i. In the optimization algorithm, the amount of calculation on each computing node does not exceed the maximum value of g_i. According to the description of the calculation amount of the node by formula (7), the relationship between the task distribution on the node p at a certain time and the calculation amount is as follows:

$$\sum_{k=1}^{\frac{m(m-1)}{2}} c_k x_{kp} \leqslant max(g_i), i = 1,2,\cdots,n \tag{14}$$

In a distributed system with n computing nodes, m data files are distributed. Each file has at most one backup on a computing node. Use y_{jp} to indicate whether to distribute the j-th data file to the p-th node. When the value of y_{jp} is 1, it means to distribute the j-th data file to the p-th node. On the node, when y_{jp} is 0, it means no distribution. Therefore, the optimal storage data distribution scheme corresponding to the data distribution strategy of the full comparison calculation under the distributed system can be described as the following equation:

$$min \sum_{p=1}^{n} \sum_{j=1}^{m} s_j y_{jp} \tag{15}$$

According to formula (13), formula (14), and formula (15), the optimized distributed system under the full comparison calculation data distribution model is shown in formula (16):

$$s.t.\begin{cases} \sum_{p=1}^{n} x_{kp} = 1, k = 1,2,\cdots,\frac{m(m-1)}{2} \\ \sum_{k=1}^{\frac{m(m-1)}{2}} c_{k=1}^{n} c_k x_{kp} \leqslant max(g_i), i = 1,2,\cdots,n \\ x_{kp} = 0or1, k = 1,2,\cdots,\frac{m(m-1)}{2}, p = 1,2,\cdots,n \\ y_{kp} = 0or1, j = 1,2,\cdots,m, p = 1,2,\cdots,n \end{cases} \tag{16}$$

According to formula (16), the data distribution of full comparison calculation is performed, and the obtained data distribution result will reduce the storage space of the distributed system, and realize complete data localization and load balancing.

3. Particle swarm optimization data distribution model algorithm design

The particle swarm algorithm is a group heuristic algorithm, which was jointly proposed by American social psychologist Kenedy and electrical engineer Eberthart in 1995. According to the theoretical analysis of the data distribution problem of the full comparison calculation above, the particle swarm optimization data distribution model consists of two parts. The first part is to find the data distribution plan and task scheduling plan that make the distributed system achieve load balance. Based on the load balancing results of the first part, the optimization solution is carried out, and the use of storage space in the distributed system is reduced under the premise of ensuring load balancing. For ease of description, the first part of the DDMPSO model is called the Data Distribution Algorithm based on Particle Swarm Optimization to achieve Load Balancing (DDAPSOLB), and the second part of the DDMPSO model is called the Data Distribution Algorithm based on Particle Swarm Optimization to Achieve Optimal Storage (DDAPSOBS).

3.1. Particle swarm optimization to achieve load balancing data distribution algorithm design

The DDAPSOLB algorithm will be designed regarding formula (13) to obtain a data distribution plan for load balancing in a distributed system. The parameter settings are shown in Table 1.

Considering that the comparison task scale of the full comparison problem will increase after the number of data files increases, and the full comparison calculation is related to the number of nodes in the distributed system, the particle dimension D is dynamically adjusted. The dimension of a single particle is specified as the product of the number of nodes in the distributed system and the number of comparison tasks t_c.

Table 1: Parameter setting

Reference	Parameter value
Maximum number of iterations T	100
Particle population size N	100
Particle dimension D	$D = n \times t_c$
Inertial weight w	[0.4,0.8]
Acceleration factor c_1	1.5
Acceleration factor c_2	1.5
Particle velocity	$[-10,10]$

Inertia weight is a very important control parameter in particle swarm algorithm, which can be used to control the development and exploration ability to solve the algorithm. In the optimization process of the DDAPSOLB algorithm, feasible solutions with better fitness values will be obtained generation by generation. Therefore, it is hoped that the particles will be more active when the optimization is started, that is, the speed will be faster. As the optimization progresses, the motion state of the particles will gradually become stable, so the linear decrementing weight strategy is adopted in the DDAPSOLB algorithm.

DDAPSOLB algorithm involves several update rules, including inertia weight update rules, particle flight speed update rules, and particle position update rules

Let w denote the inertia weight of the current particle, $w_{\min}$ and $w_{\max}$ denote the minimum and maximum values of the inertia weight respectively, $T_{\max}$ is the maximum number of iterations, and t is the current iteration number of the DDAPSOLB algorithm. Then w can be formally expressed as:

$$w = w_{max} - \frac{(w_{max} - w_{min})t}{T_{max}} \tag{17}$$

In the t-th iteration, the flight speed of the j-th dimension of particle i is$v_{ij}(t)$, c_1 and c_2 are acceleration coefficients, RAND is a random number between 0~1, and $x_{ij}(t)$ is the code of the j-th dimension of medium particle i, $p_{ij}(t)$ is the individual best position of particle i, and ,$g(t)$ is the global best position obtained in the t-th iteration. Then the flight velocity $v_{ij}(t+1)$ of the j-th dimension of particle i in the $t+1$ iteration is:

$$\begin{aligned} v_{ij}(t+1) = \quad & wv_{ij}(t) + c_1\, RAND\left[p_{ij}(t) - x_{ij}(t)\right] + \\ & c_2 RAND\left[g(t) - x_{ij}(t)\right] \end{aligned} \tag{18}$$

Whether to update the position is determined by the probability q, let the variable x_{ij} represent the probability of whether to update, when the value of x_{ij} is 1, it means updating, and when the value of x_{ij} is 0, it means not updating. Its formal description is shown in formula (19) and formula (20):

$$q = \frac{1}{(1 + e^{-v_{ij}})} \tag{19}$$

$$x_{ij} = \begin{cases} 1, r < q \\ 0, r \geqslant q \end{cases} \tag{20}$$

Among them, v_{ij} is the current velocity of the j-th dimension of particle i, and r represents a random number between 0~1. When $x_{ij} = 1$, update the position. The fitness function of the DDAPSOLB algorithm is used to evaluate the degree of load balancing of the distributed system corresponding to the particle code, and its mathematical principle follows the formula (9).

3.2. Particle swarm optimization realizes optimal storage data distribution algorithm design

The DDAPSOBS algorithm is used to optimize the amount of storage space that each node in the distributed system needs to provide under the condition of ensuring that each node in the distributed system achieves load balance. After the calculation of the DDAPSOLB algorithm is completed, a full comparison computing task scheduling scheme that enables the distributed system to achieve load balancing can be obtained, as well as the amount of calculation that each node in the distributed system needs to undertake. The maximum value of node calculation Cmax is selected from the calculation amount of each node as a constraint condition in the DDAPSOBS algorithm.

The idea of the DDAPSOBS algorithm is similar to that of the DDAPSOLB algorithm, and the relevant parameters are consistent with Table 1 except for the particle population size N. The particle population size N is taken as 10 in the DDAPSOBS algorithm. The purpose of this is to reduce the population size and improve the computational efficiency of the algorithm.

The DDAPSOLB algorithm obtains the best position of the particles that enable the system to achieve load balancing. Therefore, in the population initialization work in the DDAPSOBS algorithm, only N copies of the code are needed, and the code of the particles can be adjusted by satisfying the formula (14) to achieve the diversity of the initial population. The fitness value corresponding to the DDAPSOBS algorithm is the storage space size of the distributed system corresponding to the current code of the particle. Using the formula (16) to develop the algorithm, the update rules related to particle swarm optimization are consistent with the DDAPSOLB algorithm, and the specific location update rules have been changed.

4. Related experiments and result analysis

4.1. Evaluation index

This paper will analyze and evaluate the DDMPSO model using four evaluation indicators: load balancing, storage-saving rate, data localization rate, and model calculation time.

Load balancing. According to formula (9), the load balance deviation degree f_i of node i can be obtained, and the set of all f_i is F. Load balancing has three states: full load balancing STATE1, approximate load balancing STATE2, and non-load balancing STATE3. The mathematical description corresponding to load balancing LB is shown in formula (21), where c is the set of calculations for all tasks.

$$LB = \begin{cases} STATE\,1, max(F) - min(F) = 0 \\ STATE2, max(F) - min(F) \leqslant max(c) \\ STATE3, others \end{cases} \tag{21}$$

The calculation of the storage-saving rate will use the entire data required for the full comparison calculation to distribute a copy to each node in the distributed system. The storage space required by the distributed system is used as the denominator, and the numerator is the storage space required by each node, sum. m data files, n nodes, let h_r be the storage-saving rate, the calculation formula of h_r is as follows:

$$h_r = 1 - \frac{\sum_{i=1}^{n} \sum_{j=1}^{m'} s_{ij}}{n \sum_{k=1}^{m} s_k}, m' \leqslant m \tag{22}$$

In formula (22), s_k represents the size of file k, s_{ij} is the size of file j distributed to i, and m' represents the number of files distributed to node i.

4.2. Experimental plan design

Considering whether the file size is the same and whether the number of comparison tasks of the full comparison calculation can be divisible by the number of nodes in the distributed system, and analyzing the combination of these two conditions, the following four sets of experimental schemes can be obtained.

(1) The file size is the same, and the number of comparison tasks cannot be divisible by the number of nodes

(2) The file sizes are not the same, and the number of comparison tasks can be divisible by the number of nodes.

(3) The file size is not the same, and the number of comparison tasks cannot be divisible by the number of nodes

A gene sequence file downloaded from the National Biotechnology Center will be segmented and processed. Divide the gene sequence file into 12 small files of different sizes, and select 10 data files from them for experiment each time. The sizes of the data files after segmentation are [9.7 MB, 9.7 MB, 9.7 MB, 9.7 MB, 9.7 MB, 9.7 MB, 9.7 MB, 9.7 MB, 9.7 MB, 9.7 MB, 8.1 MB, 12.1 MB]. Take the data file as an example to design the experiment scheme.

When using Hadoop for data distribution, it has better storage-saving performance [13]. Using HDFS components and MapReduce components to cooperate, can better complete the data distribution work of the full comparison calculation. Before performing the DDMPSO model verification experiment, first, use Hadoop to perform four sets of data distribution experiments, and write a MapReduce program to read the data files to simulate the file loading operation in the sequence comparison. Record the data distribution plan and task scheduling of the full comparison calculation in each group of experiments, and use it for comparison with the experimental results of the DDMPSO model. The files and node information of the four groups of experiments are shown in Table 2. The experimental scheme shown in Table 2 is designed to cover the possible conditions of the file size being the same and whether the number of tasks can be evenly divided by nodes, and it is universal for the number of other nodes and file size lists.

Table 2: Experimental plan design

Experiment number	Number of nodes	File size list
1	5	[9.7MB,9.7MB,9.7MB,9.7MB,9.7MB, 9.7MB,9.7MB,9.7MB,9.7MB,9.7MB]
2	4	[9.7MB,9.7MB,9.7MB,9.7MB,9.7MB, 9.7MB,9.7MB,9.7MB,9.7MB,9.7MB]
3	5	[9.7MB,9.7MB,9.7MB,9.7MB,9.7MB, 9.7MB,9.7MB,9.7MB,8.1MB,12.1MB]
4	4	[9.7MB,9.7MB,9.7MB,9.7MB,9.7MB, 9.7MB,9.7MB,9.7MB,8.1MB,12.1MB]

Table 3: Hadoop experiment results

Experiment number	Node number	Data distribution plan	Task performance
1	Node1	2,3,4,5,7,9,10	37,38,39,40,41,42,43,44,45
	Node2	2,3,4,6,8,10	28,29,30,31,32,33,34,35,36
	Node3	1,4,6,7,8,9,10	1,2,3,4,5,6,7,8,9
	Node4	1,5,6,7,8,9	10,11,12,13,14,15,16,17,18
	Node5	1,2,3,5	19,20,21,22,23,24,25,26,27
2	Node1	1,2,3,4,6,7,8,9,10	37,38,39,40,41,42,43,44,45
	Node2	1,3,5,7,8,9,10	25,26,27,28,29,30,31,32,33,34,35,36
	Node3	1,2,4,5,6,10	1,2,3,4,5,6,7,8,9,10,11,12
	Node4	2,3,4,5,6,7,8,9	13,14,15,16,17,18,19,20,21,22,23,24
3	Node1	1,2,4,6,7,8,10	1,2,3,4,5,6,7,8,9
	Node2	1,2,3,5,7,10	28,29,30,31,32,33,34,35,36
	Node3	1,4,5,6,8,9	10,11,12,13,14,15,16,17,18
	Node4	2,3,5,7,8,9	19,20,21,22,23,24,25,26,27
	Node5	3,4,6,9,10	37,38,39,40,41,42,43,44,45
4	Node1	1,2,3,4,8,9,10	13,14,15,16,17,18,19,20,21,22,23,24
	Node2	1,3,5,6,7,8,10	25,26,27,28,29,30,31,32,33,34,35,36
	Node3	2,4,5,6,7,8,9,10	37,38,39,40,41,42,43,44,45
	Node4	1,2,3,4,5,6,7,9	1,2,3,4,5,6,7,8,9,10,11,12

4.3. Hadoop data distribution experiment

This article compares the data distribution results using Hadoop with the data distribution results obtained from the DDMPSO model in terms of three evaluation indicators: load balance, storage savings, and data localization.

Table 3 shows the data distribution plan of Hadoop and the execution of full comparison computing tasks obtained by completing four sets of experiments in sequence based on Hadoop. The statistics of the data distribution plan are obtained based on the web interface provided by Hadoop, and the task execution status is obtained based on the simulated full comparison calculation program.

Compared with the data distribution scheme in which all the data required for the full comparison calculation is distributed to all nodes once, the storage savings of the Hadoop cluster can be obtained. The storage-saving rate of the four sets of experiments is shown in Figure 1. In this experiment, the default number of copies of Hadoop is 3, the number of computing nodes in experiment 2 and experiment 4 is 4, and the storage-saving rate of the distributed system is 25%; when the number of computing nodes is 5, the distribution of experiment 1 and experiment 3 The storage-saving rate of the integrated system is 40%.

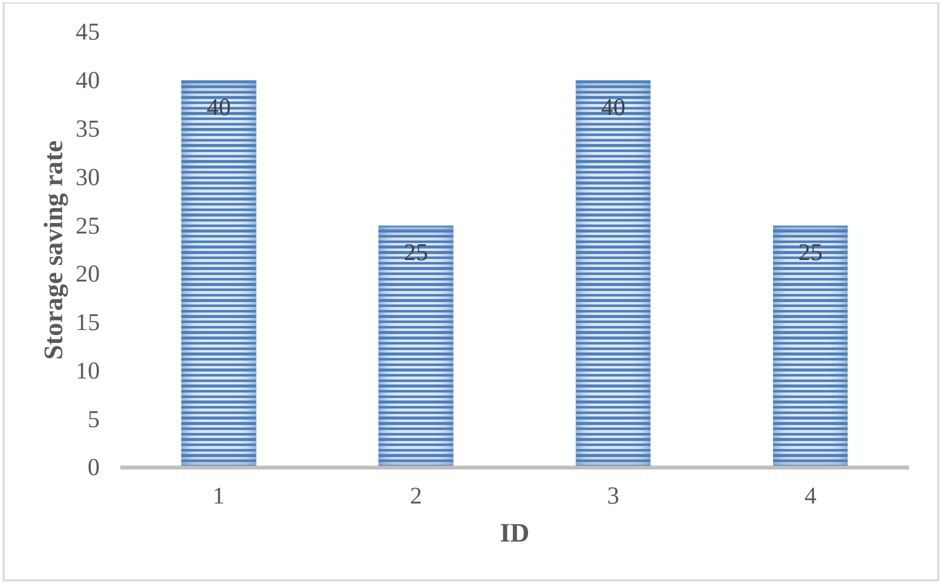

Figure 1: Storage savings rate of Hadoop experiment

Analyze the data distribution scheme and task execution in Table 3, combined with the data sending characteristics of Hadoop when requesting data from HDFS: When the client requests data from HDFS, the NameNode will send the data to the DataNode that meets the conditions and is closest to the client. The data is sent to the client. It can be analyzed whether the data requested by each comparison task is local or other nodes during the full comparison calculation. The data local rate of the node in the four experiments is shown in Figure 2. It can be seen from the figure that when Hadoop is used for full comparison calculation when the number of copies of the data file is 3, the distributed system cannot achieve complete data localization.

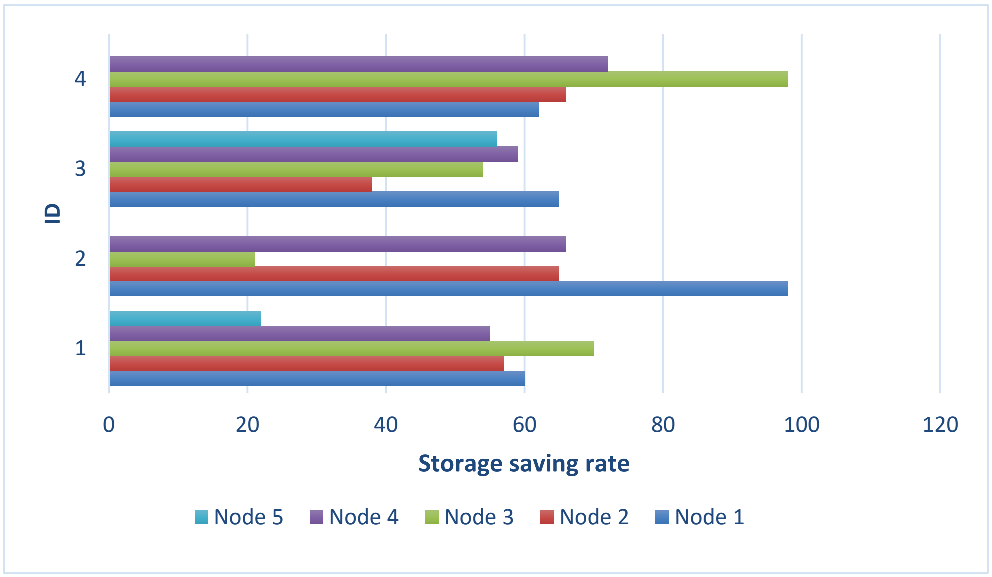

Figure 2: Data localization for Hadoop experiment

4.4. DDMPSO model experiment

Experiment with the experimental plan designed in Section 4.2. The storage-saving rate of the distributed system is shown in Figure 3, and the load balancing situation is shown in Figure 4.

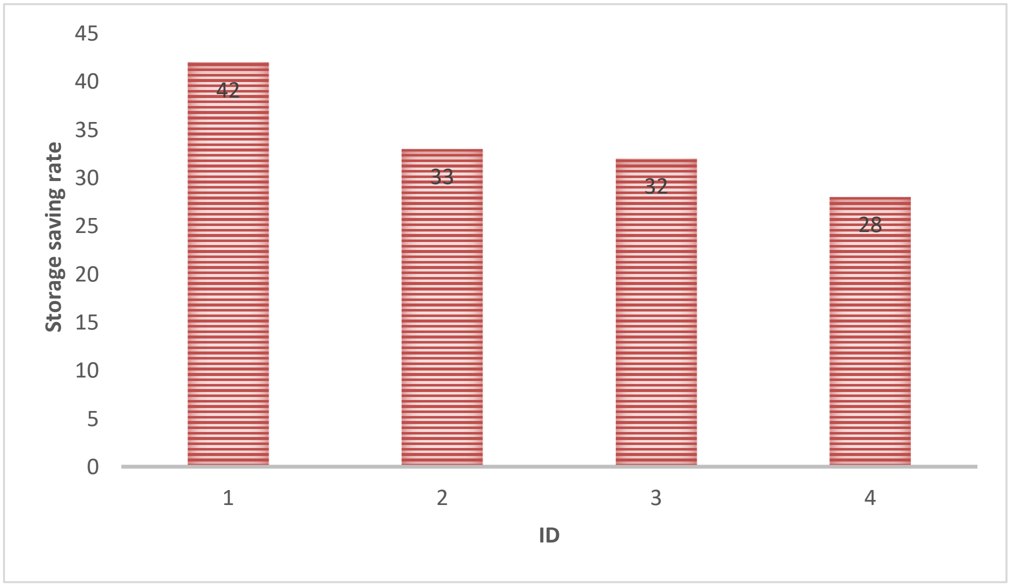

Figure 3: Storage savings for the DDMPSO model

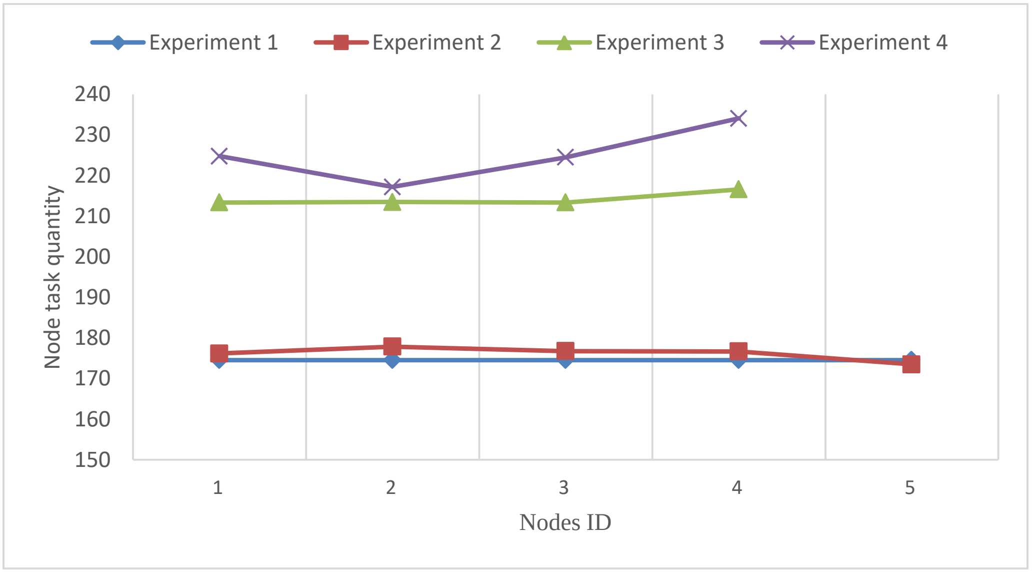

Figure 4: Load balancing of DDMPSO model

It can be seen from Figure 3 that the storage-saving rate of the DDMPSO model is higher than the data distribution strategy of Hadoop in most cases. Although the storage-saving rate of Experiment 3 is not as good as that of Hadoop, the data localization rate of the DDMPSO model can achieve complete data localization of all nodes under any circumstances, while the data distribution strategy of Hadoop cannot achieve this effect. Analysis of Figure 4 shows that the DDMPSO model can enable full comparison calculations in distributed systems to achieve complete load balancing or approximate load balancing. When the branch and bound method is used to solve the data of the same scale, it often takes 20 h to solve the problem, and the DDMPSO model can complete the calculation in 20-40 s. From an efficiency point of view, the DDMPSO model has significant advantages.

4. Conclusion

The full comparison computing data distribution strategy is the key to improving the overall computing performance of the distributed cluster system. Aiming at the disadvantages of existing data distribution strategies such as unbalanced computing load, incomplete data localization, wasted storage space, and slow computing speed, this article takes load balancing and optimized storage as the optimization goal under the premise of satisfying complete data localization. Combined with an optimized particle swarm algorithm, a data distribution model is proposed. The model optimizes the particle evolution rules in the way of task disturbance and exchange of tasks, which effectively avoids the algorithm from falling into the local optimum. Through experiments such as computational load, storage occupancy, and data localization, the results show that compared with the data distribution strategy of the open-source framework Hadoop, the data distribution model and algorithm of the proposed particle swarm optimization algorithm has computational load balancing, complete data localization, Advantages such as small storage space and fast calculation speed.

Reference

[1] T. Cole, P. Lior, and S. L. Salzberg, Bioinformatics advance access published March 16, 2009, TopHat: discovering splice junctions with RNA-Seq., Bioinformatics, vol.9, pp.9

[2] N. K. Ratha, J. H. Connell, and R. M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," IBM Systems Journal, vol.40, no.3, pp.614-634, (2001)

[3] A. Mehmet, D. Janani, and D. Dick, "Caretta: A multiple protein structure alignment and feature extraction suite," Computational and Structural Biotechnology Journal, vol.18, pp.981-992, (2020)

[4] A. Luay, A. S. Mohammed, and A. Mahmoud, "Al-sharif: A scalable multiple pairwise protein sequence alignment acceleration using hybrid CPU-GPU approach," Cluster Computing, vol.8, pp.1-12, (2020)

[5] R. C. Fong, W. J. Scheirer, and D. D. Cox, "Using human brain activity to guide machine learning," Scientific Reports, vol.8, no.1, pp.5397, (2018)

[6] M. Gardner, J. Grus, M. Neumann, "AllenNLP: A deep semantic natural language processing platform," Proceedings of Workshop for NLP Open Source Software (NLP-OSS), (2018)

[7] F Ghofrani, Q. He, and R M P. Goverde, "Recent applications of big data analytics in railway transportation systems: A survey," Transportation Research, vol.90, pp:226-246, (MAY) (2018)

[8] M. Lazri, K. Labadi, and J. Brucker, "Improving satellite rainfall estimation from MSG data in Northern Algeria by using a multi- classifier model based on machine learning," Journal of Hydrology, vol.584, pp.124705, (2020)

[9] F. M. Mendonca, "Biological sequence comparison on hybrid platforms with dynamic workload adjustment," 2013 IEEE International Symposium on Parallel &Distributed Processing, Workshops, and Ph.D. Forum, Cambridge, MA, pp.501-509, (2013)

[10] L. J. Mohan, P. Ca Neleo, U. Parampalli, "Geo-aware erasure coding for high-performance erasure-coded storage clusters," Annals of Telecommunications Annales Des Télé communications, vol.73, no.4, pp.139-152, (2018)

[11] X. D. Meng and V. A. Chaudhary, "A high-performance heterogeneous computing platform for biological sequence analysis," IEEE Transactions on Parallel and Distributed Systems, vol.21, no.9, pp.1267-1280, (2010)

[12] S. Tiwari, K. Kaur, and Y. Pathak, "Computed tomography reconstruction on distributed storage using hybrid regularization approach," Modern Physics Letters B, (2019)

[13] V. Trikha, J. Fiaidhi, and S. Mohammed, "Identifying EEG Binary Limb Motor Imagery Movements using Thick Data Analytics," Asia-pacific Journal of Convergent Research Interchange, FuCoS, ISSN: 2508-9080 (Print); 2671-5325 (Online), vol.6, no.9, September, pp.169-189, (2020), DOI:10.47116/apjcri.2020.09.15

[14] J. Mononteliza, "Research on IoT reservation algorithm based on deep learning," Asia-pacific Journal of Convergent Research Interchange, FuCoS, ISSN: 2508-9080 (Print); 2671-5325 (Online), vol.6, no.9, September, pp.191-205, (2020), DOI:10.47116/apjcri.2020.09.16

[15] T. Sai Raaga Sowmya, “Cost minimization for big data processing in geo-distributed data centers,” Asia-pacific Journal of Convergent Research Interchange, SoCoRI, ISSN: 2508-9080 (Print); 2671-5325 (Online), vol.2, no.4, December, pp.33-41, (2016), DOI:10.21742/APJCRI.2016.12.05

[16] L. Meng, S. Zhang, and F. Wang, “Influence of internet-based social big data on personal credit reporting,” Asia-pacific Journal of Convergent Research Interchange, FuCoS, ISSN: 2508-9080 (Print); 2671-5325 (Online), vol.6, no.7, July, pp.39-57, (2020), DOI:10.47116/apjcri.2020.07.05

[17] X. Wang, “The construction of employment and entrepreneurship service system for university students based on big data,” Asia-pacific Journal of Convergent Research Interchange, FuCoS, ISSN: 2508-9080 (Print); 2671-5325 (Online), vol.6, no.10, October, pp.203-213, (2020), DOI:10.47116/apjcri.2020.10.16

[18] K. B. Pierce, J. L. Ohmann, and M. C. Wimberly, "Mapping wildland fuels and forest structure for land management: A comparison of nearest neighbor imputation and other methods," Revue Canadienne De Recherche Forestière, vol.39, no.10, pp.1901-1916, (2009)

[19] J. Kennedy and R. C. Eberhart, “Swarm intelligence,” USA: Academic Press, (2001)