An Improved Dynamic Load Balancing Optimization Algorithm Based on CS

XiuCai Guo, Yue Zhang* and YaoYi He

School of Xi'an University of Science and Technology, Beijing, China

*Corresponding author e-mail: m17749133057@163.com

Abstract. Aiming at the problem that the existing load balancing algorithm has a long response delay and a low response connection ratio, which cannot meet the timely processing of large amounts of data in the existing application process, this paper proposes a weighted minimum connection algorithm (CS-WLC). The algorithm utilizes the optimization feature of the Cuckoo Search Algorithm (CS) to classify the server first. Because the algorithm requires fewer parameters and the optimization rate is high, the time for selecting the minimum number of connections is reduced. The experimental results show that under the same conditions, the improved algorithm has smaller response delay and higher throughout capacity, and has better load balancing effect.

1. Preface
In recent years, the continuous development of information technology, coupled with the application of cloud platforms in various industries, will create connections between different systems. As the number of users continues to increase, the amount of data in the system is increasing dramatically. [1, 3] At this time, it is necessary to solve the data processing problems caused by various problems such as large amounts of data and heterogeneous data. [2, 4] A single server can no longer support the current amount of data processing requests and cannot meet the needs of actual production. This requires multiple servers to run simultaneously, introduces load balancing, and reasonably allocates requests to each server for processing. [5]

There are two types of load balancing: dynamic and static. Static tasks are assigned based on a fixed ratio, while dynamic tasks are assigned based on the current server status. However, due to the complexity of data types in the existing system, existing algorithms cannot effectively process the data in the actual application process, which results in a slow data processing speed and the inability to reasonably use existing resources to complete scheduling tasks. [6] Resulting in security risks and waste of resources. Based on an in-depth understanding and research of the cuckoo search algorithm and weighted least connection method, this paper proposes a weighted least connection algorithm (CS-WLC) based on cuckoo search. Before comparing the current load of all server nodes, first classify the server nodes and find the optimal solution in each group of servers. [7] Next, combine all the optimal solutions into a new set. Use the least connection method to minimize The number of connections is selected, which reduces system delay and improves system resource utilization.
2. Weighted minimum connection method

The Weighted Least-Connection algorithm is an extension of the least connection method. Its core idea is as follows:

Suppose there is a server $S_1, S_2, \ldots, S_n$, the weight of the server $S_i$ is $W(S_i)$, the current number of connections is $C(S_i)$, then $C(\text{sum}) = \sum C(S_i) = 1, 2, \ldots, n$. When the server $S_m$ satisfies the formula (1), a new connection request is sent to the server $S_m$.

$$\frac{C(S_m)}{W(S_m)}/C(\text{sum}) = \min \left\{ \frac{C(S_i)}{W(S_i)} \right\}$$

Among them $i = 1, 2, \ldots, n, W(S_i) \neq 0$.

As in the actual operation process, $C(\text{sum})$ is a constant, then equation (1) can be simplified to the following formula:

$$\frac{C(S_m)}{W(S_m)} = \min \left( \frac{C(S_i)}{W(S_i)} \right)$$

Among them $i = 1, 2, \ldots, n, W(S_i) \neq 0$.

In the above algorithm flow, each server expresses its performance with a corresponding weight. When assigning a new connection request, the server weight is made as proportional as possible to the number of established connections. Because the algorithm uses a single number of connections as an indicator, the accuracy is not high. Therefore, further introduction of server categories, CPU usage, network bandwidth occupancy and other factors is needed for comprehensive consideration [7].

3. Improved load balancing mechanism

3.1. Cuckoo algorithm

The CS algorithm evolved based on the cuckoo's nest-finding method. Cuckoos used a random method to find a suitable bird's nest. The main method is to generate candidate birdhouses to keep during flight and keep updating them, and finally make cuckoos find the best birdhouse locations. The formula for nest search path and location update is as follows:

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \otimes L(\lambda)$$

The position of the bird's nest in the $i$ nest in the t-th step is denoted by $x_i^{(t+1)}$, $\alpha$ is the step size, and $L(\lambda)$ is the random search path. When the position of the bird's nest is updated, the random number $r$ of the uniformly distributed $[0,1]$ is compared with $Pa$. If $r > Pa$, $x_i^{(t+1)}$ is updated to find a new optimal solution, and otherwise the current optimal bird's nest position is retained.

3.2. Improved load balancing algorithm

The basic idea of the algorithm in this paper is: Before the original weighted minimum connection method is used to select the minimum connection number node, the cuckoo search algorithm is introduced. Firstly, the idea of the cuckoo algorithm is applied, and the better solution set $Pr = \{P(S_1), P(S_2), \ldots, P(S_m)\}$ is selected. Then, according to the basic steps of the weighted minimum connection method, the next step is performed. The calculation is performed in the better solution set,
so that the server node \( P(S_i) \) with the smallest number of connections is obtained, and the current task is matched, thereby completing the load distribution task.

### 3.2.1. Specific algorithm flow

After a long period of operation of the computer cluster system, the load recorded on the scheduler will not accurately reflect the real situation of the load of each service node in real time. Therefore, it is necessary to periodically collect the load information of the service node, thereby ensuring the accuracy of the data recorded in the load balancing server. That is, every other time \( T \), each service node feeds back five performance parameters of the CPU utilization, memory utilization, disk access rate, network bandwidth occupancy, and process occupancy rate of the node to the load balancing server.

#### Maximum processing power of each server node

Assume that the number of CPUs of the \( i \)-th server node is \( n_i \), the processing rate is \( C_i \), the memory capacity is \( M_i \), the disk IO rate is \( D_i \), the network throughput is \( N_i \), and the maximum number of processes is \( P(S_i) \). The maximum processing capacity of the server node is:

\[
W_i = K_1 \cdot n_i \cdot C_i + K_2 \cdot M_i + K_3 \cdot D_i + K_4 \cdot N_i + K_5 \cdot (Pr_i) \tag{4}
\]

Among them \( \sum_{K=1,i=1,2,\cdots,n} \).

#### Current utilization of each server node

Assuming that the number of CPUs of the \( i \)-th server node is \( n_i \), and the real-time processing rate, memory capacity, disk IO rate, network throughput, and number of processes are respectively \( C_i, M_i, D_i, N_i, P(S_i) \), the real-time processing capability of the server node is:

\[
W_i = K_1 \cdot n_i \cdot C_i + K_2 \cdot M_i + K_3 \cdot D_i + K_4 \cdot N_i + K_5 \cdot (Pr_i) \tag{5}
\]

Among them \( \sum_{K=1,i=1,2,\cdots,n} \).

The current utilization of each server node is:

\[
P_i = \frac{W_i}{W} \times 100\% \tag{6}
\]

#### Set the reference value

Set the reference value \( P(S) = [P(S_1), P(S_2), \ldots, P(S_n)] \). When \( P_i \leq P(S_i) \), put the Pi value less than P(S1) into P(S1), \( P(S_1) = [P_1^{(1)}, P_1^{(2)}, \ldots, P_1^{(n)}] \); at that time, put the Pi value of \( P(S_i) \leq P_i \leq P(S_j) \) greater than P(S1) less than P(S2) into P (S2), \( P(S_2) = [P_2^{(1)}, P_2^{(2)}, \ldots, P_2^{(n)}] \).

And so on, when \( P(S_{n-1}) \leq P_i \leq P(S_n) \), \( P(S_m) = [P_m^{(1)}, P_m^{(2)}, \ldots, P_m^{(n)}] \).

#### Introducing the cuckoo search algorithm

After sorting the real-time load of the server node in the previous step, use the cuckoo search algorithm for P (S1), P(S2), ..., P(Sm) to perform global search to obtain the optimal value. As follows: Step 1: Assume that the number of server nodes is \( n \), the search space dimension is \( d \), initialize the current load utilization of the server node, \( P(0) = [P_1^{(0)}, P_2^{(0)}, \ldots, P_n^{(0)}] \), and find the current optimal server node, that is, the server node with the smallest current load. Its current load is \( P_i^{(0)}, i \in [1, 2, \cdots, n] \).

Step 2: Loop body

Location update

Keep the server node \( P(t-1) \) with the minimum utilization of the previous step, \( t \) is an integer, and updates other nodes according to formula (1) to obtain a new set of nodes, and tests the new group
with the previous group. $P(t-1) [p_1^{(t-1)}, p_2^{(t-1)}, \ldots, p_n^{(t-1)}]$ compares and replaces the nodes with higher utilization rate with nodes with lower utilization rate, thereby obtaining a better set of server node groups $P(t) [p_1^{(t)}, p_2^{(t)}, \ldots, p_n^{(t)}]$.

2) Deposit insurance

The security will be subject to the evenly distributed random number $r \in [0, 1]$, compared with $P_0$, retain the nodes with less utilization in $P(t)$, and randomly change the nodes with higher utilization to obtain a new set of data, the group. The data is tested, compared with the node data in $P(t)$, and the node with better test value is substituted for the worse node, and a new set of better node data is obtained $P(t+1) [p_1^{(t+1)}, p_2^{(t+1)}, \ldots, p_n^{(t+1)}]$.

Step 3: Find the optimal Node $P^{(t+1)}$ in $P(t+1)$.

After the global optimization of the cuckoo search algorithm, a new set of solutions will be obtained, each of which is the optimal value of the respective interval, and these nodes form a new node group $W(t) [p_1^{(t)}, p_2^{(t)}, \ldots, p_n^{(t)}]$.

The minimum connection method selects the final node.

Because within the cycle time $T$, each server node gets different number of request connections due to different weights, the greater the weight, the more the number of connections to the node is requested, the greater the server node usage rate. When the request arrives, the single connection number does not reflect the load receiving capability of the current node well. Therefore, each node is assigned the corresponding weight first, and then the minimum connection method is used to select, and finally a current optimal is obtained. Solution, assigning a server request to the node.

Assume that the weight of the $j$th server node is $W(S_j)$, and the number of connections of each node is $P^{(t)}$.

$$P(sum) = P^{(1)} + P^{(2)} + \cdots + P^{(n)} = \sum_{j=1}^{n} P^{(j)}$$  \hspace{1cm} (7)

When the $P^{(i)}$ server satisfies the formula (8), when a new request arrives, the request is assigned to $P^{(i)}$.

$$\frac{P^{(i)}}{W(S_i)} \cdot P(sum) = \min \left\{ \frac{P^{(j)}}{W(S_j)} \right\}$$  \hspace{1cm} (8)

Among them, $i = 1, 2, \ldots, n$, $i \in \{1, 2, \ldots, n\}$, $W(S_j) \neq 0$.

Because $P(sum)$ is a constant during the calculation, the above formula can be simplified as:

$$\frac{P^{(i)}}{W(S_i)} = \min \left\{ \frac{P^{(j)}}{W(S_j)} \right\}$$  \hspace{1cm} (9)

Among them, $i = 1, 2, \ldots, n$, $i \in \{1, 2, \ldots, n\}$, $W(S_j) \neq 0$.

4. Improved algorithm verification

In this experiment, response delay and throughput are used as evaluation indicators to improve the performance of the algorithm. Nginx is used to build a web server cluster, and the Apache test tool ApacheBench is used to test the performance of the system. [3] The hardware equipment conditions used in the experiment are shown in Table 1. During the test, $P_a = 0.25$ in the cuckoo search algorithm differs depending on the CPU processing rate, memory capacity, disk IO rate, network throughput, and
number of processes. The impact of each indicator on the system varies. Therefore, in the experiment, the \( K = \{K_1, K_2, K_3, K_4, K_5\} = \{0.3, 0.2, 0.2, 0.2, 0.1\}\) is taken separately, and the weight of each server in the experiment is taken as \( W(S) = [W(S1), W(S2), \ldots, W(S5)] = \{0.3, 0.2, 0.3, 0.1, 0.1\}\). Under the same conditions, the performance of the weighted minimum connection method and the improved algorithm (CS-WLC) are analyzed to obtain the corresponding response delay and throughput. The experimental results are as follows.

**Table 1. Basic conditions of hardware equipment**

| Device No | CPU frequency | Memory Capacity | System type | Equipment use            |
|-----------|---------------|-----------------|-------------|--------------------------|
| 1         | 3.2GHz        | 16GB            | 64-bit      | Load balancing server    |
| 2         | 3.2GHz        | 4GB             | 64-bit      | Backend server           |
| 3         | 3.2GHz        | 16GB            | 64-bit      | Backend server           |
| 4         | 2.16GHz       | 8GB             | 64-bit      | Backend server           |
| 5         | 2.16GHz       | 8GB             | 64-bit      | Backend server           |

**Table 2. Response delay and response connections of the two algorithms**

| Number of connections | Response delay (ms) | Number of response connections |
|-----------------------|---------------------|--------------------------------|
|                       | Weighted minimum connection method | Improved algorithm in this paper | Weighted minimum connection method | Improved algorithm in this paper |
| 50                    | 2.54                | 2.58                           | 50                             | 49                               |
| 100                   | 2.59                | 2.61                           | 98                             | 99                               |
| 300                   | 2.65                | 2.66                           | 298                            | 297                              |
| 500                   | 3.34                | 3.23                           | 483                            | 490                              |
| 1000                  | 135.42              | 117.82                         | 927                            | 979                              |
| 1500                  | 453.60              | 390.87                         | 1301                           | 1420                             |

**Figure 1.** Comparison of two algorithms’ response delay time
Figure 2. Comparison of two kinds of algorithm response connections

It can be seen from the experimental results that the improved algorithm has obvious advantages in the case of increasing number of connections. When the number of connections is 1000, the response delay is increased by 12.997% compared with the weighted minimum connection method, and the number of response connections is increased by 5.61%. When the number of connections is 1500, the response delay is 13.829% higher than the weighted minimum connection method, and the response connection number is increased by 9.15%. According to the trend, the improved algorithm (CS-WLC) in this paper has better performance under the situation of increasing number of connections, and it is more suitable for application systems with large data volume.

5. Overall summary
This paper introduces the weighted least connection method (WLC) and cuckoo search algorithm (CS), and proposes a weighted least connection algorithm (CS-WLC) based on cuckoo search. The response delay and throughput are used as performance indicators. After experimental verification, it is found that the response delay of the improved algorithm (CS-WLC) is smaller than the weighted minimum connection method (WLC) under the same conditions, and the number of response connections is more. Therefore, we know that the improved algorithm (CS-WLC) in this paper has better load balancing effect.

References
[1] He Yaoyi, Wang Haibo. Research on the fusion of coal mine monitoring system based on Internet of Things [J/OL]. Industrial and mining automation: 1-7[2019-08-21].https://doi.org/10.13272/j. Issn.1671-251x.17458.
[2] Sun Jiping, Zhang Gaomin. Mine emergency communication system [J/OL]. Industrial automation: 1-6[2019-08-21].https://doi.org/10.13272/j.issn.1671-251x. 17483.
[3] Qiuyan Pei. Research on data mining algorithm and intelligent analysis system for massive ventilation in coal mines [D]. Taiyuan University of Technology, 2017.
[4] Gao Wen. Remote fault analysis platform for coal mine monitoring system [J]. Industrial and mining automation, 2019, 45 (02): 105-108.
[5] Lu Wei, Hao Jianjun, Ding Xiaoming. Timing Synchronization Algorithm for OFDM Symbols in Mine Through-ground Communication[J]. Industrial Automation, 2015, 41(08): 56-59.
[6] Wang Peng, Huang Hongqiong. Improved load balancing algorithm based on neural network feedback mechanism [J]. Modern Computer (Professional Edition), 2018 (12): 3-9.
[7] Chen Taian. An improved dynamic feedback load balancing algorithm [J]. Electronic Design Engineering, 2013, 21 (05): 171-172+179.