U.P.B. Sci. Bull., Series C, Vol. 84, Iss. 2, 2022 ISSN 2286-3540

ELEPHANT FLOW SCHEDULING IN SDN DATA CENTER NETWORK BASED ON DIFFERENTIAL EVOLUTION ALGORITHM

Rongrong DAI[1], Honghui LI[2]*, Xueliang FU[3]

The traditional traffic scheduling method when scheduling the elephant flow may cause network congestion and link load imbalance. A new mechanism DE-SDN based on differential evolution is proposed to optimize the scheduling of the elephant flow in the SDN data center network, and the optimization objective is to minimize the maximum link utilization. By defining differential mutation, crossover and selection operations, and combining with the global network state, the algorithm can obtain the optimal path and reroute the elephant flow on the congested link. The experiments show that compared with ECMP and GFF, taking random communication mode as an example, the proposed DE-SDN algorithm improves the network pairwise bandwidth by 16.74%~32.09% and 7.95%~23.62% respectively, reduces the maximum link utilization rate and achieves the network load balancing well.

Keywords: Data center network, Software defined network, Traffic scheduling, Differential evolution algorithm

1. Introduction

In recent years, big data and icloud computing continue to develop, data centers constructed by two or three-layer switches or routers have become the infrastructure of Internet information construction [1]. As the scale of the data center is gradually growing, the number of communications within the data center network has grown index, and its bandwidth demand is increasing [2]. Traditional data center network structure is overburdened, which is prone to link congestion and can’t provide effective traffic transmission services. Therefore, network traffic scheduling problems has attracted more and more attention.

The dynamic information of the whole network cannot be collected and understood by routing algorithms in the traditional data center network architecture, and the global optimization scheduling of network traffic cannot be achieved. And software defined networks (abbreviated SDN [3]) separates the

[1] MA., College of Computer and Information Engineering, Inner Mongolia Agricultural University, Hohhot, China, e-mail: gegenrlmc@163.com

[2] Prof., College of Computer and Information Engineering, Inner Mongolia Agricultural University, Hohhot, China, e-mail: lihh@imau.edu.cn

[3] College of Computer and Information Engineering, Inner Mongolia Agricultural University, Hohhot, China.

control plane and the data plane of the switch, and the controller can master the use of the whole network in real time, realize the scheduling of network traffic more accurately, and provide a new solution for the development of new network applications and future Internet technology [4].

So far, Equal-Cost Multi-Path Routing (abbreviated ECMP) is extensively applied in data center network traffic scheduling [5]. When there are multiple equivalent available paths to the same destination node, the ECMP algorithm hashed each data flow and evenly distributed the data flows to the multiple equivalent paths according to the hash value. In this way, the network load balancing is realized. Research show that current data center network traffic is divided into two types: elephant flow and mouse flow [6]. Among them, data flows that occupy 10% or more of the link bandwidth are called elephant flows. Although elephant flow accounts for a small amount in the data center network, it lasts for a long time and carries up to 90% of the data volume [7]. ECMP algorithm is a static traffic scheduling algorithm, which does not consider the real-time network usage state. When elephant flows occur on the network, ECMP may allocate multiple data flows to the same path, resulting in link congestion, load imbalance, and low network resource utilization.

Aiming at the problem of elephant flow scheduling in ECMP algorithm, a new elephant flow scheduling mechanism DE-SDN based on differential evolution algorithm is proposed in this paper to realize global dynamic traffic scheduling. Based on ECMP algorithm, the DE-SDN algorithm monitors the link usage status and calculates the global optimal route after reroute for elephant flows on congested links. Simulation results show that compared with ECMP algorithm and GFF algorithm, DE-SDN algorithm can effectively reduce the maximum link utilization and improve the network bandwidth.

Firstly, this paper introduces the related work to solve the traffic scheduling problem of SDN data center network, and then carries out mathematical modeling of the problem. Then, a traffic scheduling method de-SDN based on differential evolution algorithm is designed to solve the optimization model. The last, the effectiveness and feasibility of the proposed method can be evaluated by simulation experiments.

2. Related work

The traditional data center traffic scheduling method has been unable to meet the rapid development of the current network demand. Many scholars proposed the traffic scheduling method of SDN data center network.

Al-fares et al. proposed the Hedera dynamic traffic scheduling mechanism [8], which was applied to the multilevel switch topology of SDN data center network. It applies the Global First Fit algorithm (abbreviated GFF) to calculate the path without conflicts and issues instructions to the switch to reroute the data flow. This is the first time that SDN technology is applied to data center network traffic scheduling, which provides a new research direction for the following traffic scheduling research.

Andrew R. Curtis et al. [9] proposed the Mahout method, which identifies elephant flows by detecting the connection cache of the terminal host rather than on switches in the network. [10] Mahout reduced switch waste compared to Hedera and was an order of magnitude faster at detecting elephant flows than all flow statistics.

Liu Zhenpeng et al. [11] proposed a dynamic traffic scheduling scheme DTSNL based on network load in view of the problem that ECMP algorithm does not consider traffic characteristics and is prone to link conflict. [12] Experimental results show that DTSNL can improve the load balancing effect of data center network compared with ECMP and GFF.

Peng Daqin et al. [13] proposed a multipath routing algorithm based on the real-time status of links to schedule elephant flow and mouse flow separately, routing elephant flow according to the path weight, and routing mouse flow by selecting the path with the maximum available remaining bandwidth. The algorithm can improve the average link utilization of Fat-Tree networks.

In recent years, with the development of swarm intelligence algorithm, researchers have applied it to solve the traffic scheduling optimization problem of SDN network and improved the network performance. Lin Zhihua et al. [14] proposed DPSOFS, a discrete particle swarm optimization traffic scheduling algorithm in the fat-Tree data center network topology. DPSOFS algorithm accelerates the convergence speed and obtains a better traffic scheduling path.

Li Honghui et al. [15] is proposed based on ant colony algorithm of elephants flows scheduling algorithm of ACO - SDN, by modeling the elephant flow scheduling problem and solving the optimization model based on the ant colony algorithm, the global optimal path is obtained to re-route the elephant flow, which effectively reduces the maximum link utilization rate of the data center network and improves the bandwidth of network allocation. Therefore, this paper proposes a traffic scheduling mechanism DE-SDN based on differential evolution algorithm to realize dynamic scheduling of elephant traffic in SDN data center network. DE-SDN algorithm calculates the global optimal path according to the real-time status of the network from the perspective of the global network and reroute elephant flow on the congested link based on ECMP. In this way, network load balancing is implemented, and network resource utilization is improved.

3. Modeling traffic scheduling problems

In this paper, the traffic scheduling problem is modeled. The main feature of the model is that under the given network topology and link capacity constraints, the network traffic is reasonably scheduled to optimize its objective function, and then it is evenly distributed on each link in the network to achieve load balance. The specific modeling description is as follows:

The data center network topology is represented as a graph $G(S,L)$, where S is the set of all network nodes (switches), and the node is $s_i \in S$, $i=1,2,......,|S|$. L indicates the set of all links in the network. The capacity of link $l \in L$ is C_l and the link utilization rate is u_l . L_i and $L_i^{'}$ are subsets of link set L respectively. Any $l_i \in L_i$ ($l_i^{'} \in L_i^{'}$)has s_i as its endpoint and data flows into ($l_i^{'}$ flows out of) nodes s_i through l_i . The set of data flows causing congestion is represented as E , the bandwidth of data flow $e \in E$ is b_e , and the source node and destination node of data flow e are represented by s_o^e and s_d^e respectively. Variable y_l^e indicates whether data flow e passes through link l . Link utilization rate u_l is defined as the ratio of the sum of bandwidths of all data flows e passing through link l to the capacity C_l of link l . Maximum link utilization $u^{\max}$ refers to the maximum value of utilization in all links in the network. So the optimization goal of the flow scheduling problem is to minimize the maximum link utilization in the network, which can be represented by the formula (1).

$$\min \qquad \max \left\{ \frac{\sum_{e \in E} y_l^e b_e}{C_l} \right\}_{l \in L} \tag{1}$$

When the above optimization target is reached, the flow rate schedule should satisfy the following constraint conditions (2) ~ (6).

$$\sum_{e \in E} y_l^e b_e \le C_l \qquad l \in L \tag{2}$$

$$\sum_{l_i \in L_i} y_l^e b_e - \sum_{l_i \in L_i^{'}} y_l^e b_e = -b_e \qquad s_i = s_o^e, e \in E \tag{3}$$

$$\sum_{l_i \in L_i} y_l^e b_e - \sum_{l_i \in L_i^{'}} y_l^e b_e = b_e \qquad s_i = s_d^e, e \in E \tag{4}$$

$$\sum_{l_i \in L_i} y_l^e b_e = \sum_{l_i \in L_i^{'}} y_l^e b_e \qquad s_i \in S - \{s_o^e, s_d^e\} \tag{5}$$

$$y_l^e \in \{0,1\} \qquad l \in L, e \in E \tag{6}$$

Formula (2) indicates that the sum of bandwidth of data flow e passing through link l cannot exceed the capacity of this link. Formula (3) indicates that if the node s_i is the source node of the data flow e , the data flow e only flows out of this node. And formula (4) indicates that if the node s_i is the destination node, the data flow e only flows into this node and no longer flows out. Formula (5) indicates that if s_i is the intermediate node of e, the data flows in and out of this node are the same. Formula (6) defines the value range of variable y_l^e.

The essence of the model in this paper is an integer linear programming mathematical model, which has obvious effects and certain advantages when there are many elephant flows in the network. In this paper, differential evolution algorithm is applied to solve the model and obtain the routing scheduling scheme of elephant flow.

4. Differential evolution traffic scheduling algorithm DE-SDN

4.1 DE-SDN Overview

The flow of the traffic scheduling mechanism DE-SDN proposed in this paper based on differential evolution algorithm is demonstrated in Fig. 1, and the specific steps are as follows.

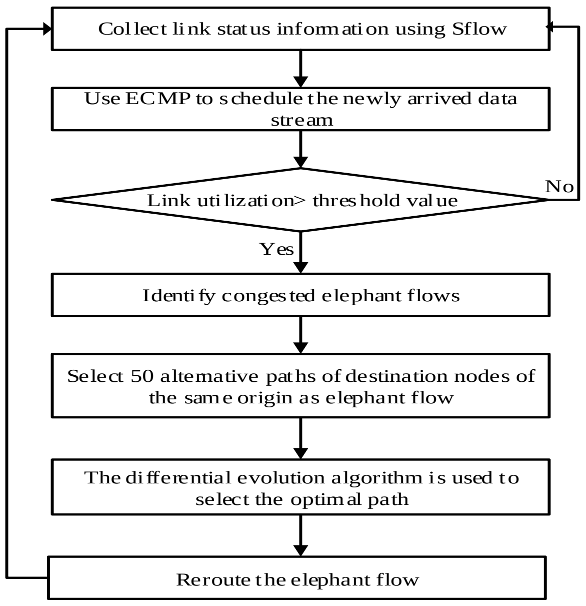

Collect link status information using Sflow
Use ECMP to schedule the newly arrived data stream
Link utilization> threshold value
No
Yes
Identify congested elephant flows
Select 50 alternative paths of destination nodes of the same origin as elephant flow
The differential evolution algorithm is used to select the optimal path
Reroute the elephant flow

Fig. 1. Elephant flow scheduling flowchart

(1) The Sflow-rt collector continuously collects data center network link status information.

(2) For the new arrival data center network data flow, the first use of ECMP algorithm flow scheduling.

(3) If the elephant flows through one of the links the maximum link utilization is greater than the threshold, then perform the following steps (4) ~ (6), or turn to step (1).

(4) K shortest available paths of the same destination node as the elephant flow are selected as alternative paths for rerouting the elephant flow.

(5) The traffic scheduling method based on differential evolution algorithm is called (see section 3.2 for details). According to the current network state collected by the Sflow-rt collector, the global optimal path is calculated from k alternative paths.

(6) Convert the calculated new path into a flow entry and send it to each switch. Then re-route the elephant flow and go to Step (1).

4.2 Differential evolution scheduling part

In 1997, Rainer Storn and Kenneth Price proposed Differential Evolution (abbreviated DE) after discovering the convergence of genetic algorithms [16]. The basic idea is to carry out mutation operation on the randomly generated population. According to certain rules, the difference vector of two individuals is summed up with the third individual to produce mutation individuals [17]. Then the mutant individual and the target individual are crossed, by calculating the fitness value and iterative calculation, in accordance with the natural law of survival of the fittest, finally select the global optimal solution. DE algorithm has fast convergence speed, few control parameters, and the algorithm is simple and easy to execute. The optimization results are more stable than genetic algorithm and particle swarm optimization algorithm [18]. According to the network topology and the current link utilization rate, a global optimal path can be calculated, and the path can also satisfy the constraint conditions (2)~(6). The input is the current utilization of links in the data center network, and the output is the optimal path of a rerouted elephant flow. The flow chart of this method is shown in Fig. 2, and the specific steps are described below.

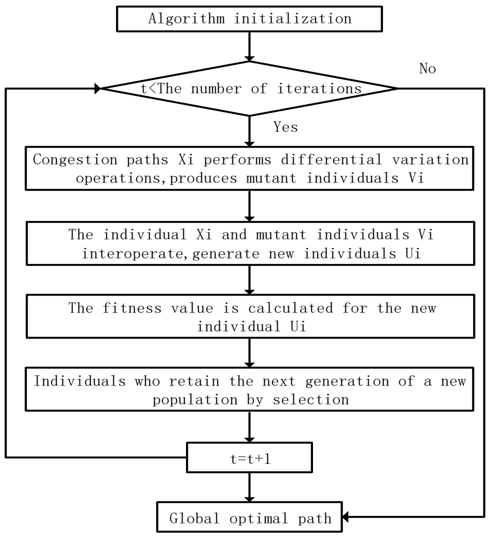

Algorithm initialization
t<The number of iterations
No
Yes
Congestion paths Xi performs differential variation operations, produces mutant individuals Vi
The individual Xi and mutant individuals Vi interoperate, generate new individuals Ui
The fitness value is calculated for the new individual Ui
Individuals who retain the next generation of a new population by selection
t=t+1
Global optimal path

Fig. 2. DE-SDN Algorithm flow chart

(1) Initialization of the algorithm

① Set loop control conditions. Set the loop control variable *t* as 1 and set the maximum number of iterations *MaxT*.

② The initial solution space *R*. According to Yen algorithm [19], *k*-shortest path is generated as an alternative path candidate solution space *R*.

③ Population initialization. *M* individuals are selected from solution space *R* in turn to form a population *Pop*, where the individual X_i is a complete path, and each path is composed of n links l, as shown in Equation (7).

$$X_i = \left\{ l_1, \ l_2, \ \dots , \ l_n \right\} \qquad X_i \in R, \ 1 \le i \le M \tag{7}$$

④ Definition of fitness function.

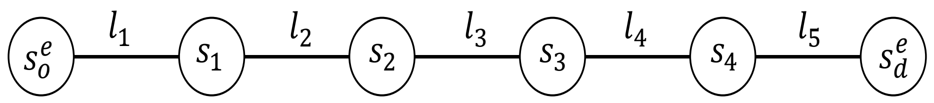
In order to achieve the optimization goal of minimizing maximum link utilization, the fitness function of individual X_i is defined as shown in Equation (8).

$$F(X_i) = 1 / \left(\mathrm{a} \times HOPS(X_i) + \mathrm{b} \times MAXU(X_i) \right) \tag{8}$$

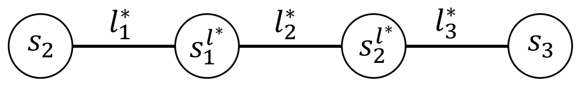
Where, $HOPS(X_i)$ is the length (hop number) of individual path X_i, $MAXU(X_i)$ is the maximum link utilization of all links in X_i, *a* and *b* are the influence factors. Through the definition of fitness function, the minimization problem solved in this paper is transformed into the maximization problem, and the optimal solution is obtained according to the following steps.

(2) Mutation operation

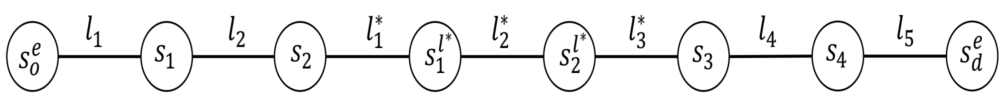
In the variation operation of the classical differential evolution algorithm, three individuals are randomly selected from the population *Pop* to generate the variation [16]. In this paper, in order to maintain the continuity of candidate paths, the link with the maximum link utilization greater than threshold *H* in individual X_i is replaced by one or several adjacent and uncongested links to form a new path individual $V_i(t)$ in the t iteration. Fig. 3 shows an example of a mutation operation.



(a) A complete path $X_i(t)$



(b) Any candidate path that satisfies the conditions l^*



(c) Variation of path $V_i(t)$

Fig. 3. Variation example

In Fig. 3, s_i represents the switch node and l_j represents the link between the two switches. Assume that the utilization of connection link l_3 on the path shown in Fig. 3(a) is greater than the threshold, then replace it. Using Yen algorithm, two endpoints s_2 and s_3 of link l_3 were taken as source and destination nodes, respectively, to generate k-shortest paths and reserve them to the mutation candidate solution space R'. Candidate path l^* satisfying the following conditions is selected from solution space R', as shown in Fig. 3(b):

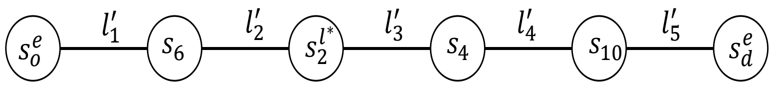
A. The maximum link utilization of all the links in the candidate path l^* is lower than the Threshold H .

B. The candidate path l^* is the shortest in the solution space R'. Then, the congested link l_3 was replaced by the candidate path l^* to generate the variation individuals $V_i(t)$ as shown in Fig. 3(c).

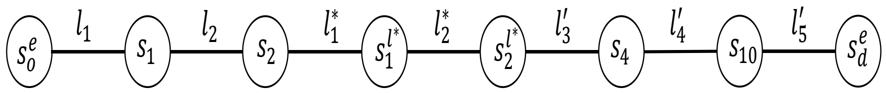
(3) Cross operation

In iteration *t*, a random number r is selected from the interval [0, 1]. If $r >$ crossover factor cr, the mutant individual $V_i(t)$ is retained and the step (4) is carried out. If $r \leq$ crossover factor cr, crossover operation was performed between the original individual $X_i(t)$ and the mutant individual $V_i(t)$. The individual $X_i(t)$ and $V_i(t)$ are traversed sequentially. When the first same node (common node) appears, all links after this node are exchanged to obtain the crossed individual $U_i(t)$. Then determine whether there is a loop in $U_i(t)$. If so, retain the path to remove the loop as the crossed individual $U_i(t)$.

Considering that the mutant individual $V_i(t)$ may be derived from the variation of $X_i(t)$ (as mentioned above), the crossover individual $U_i(t)$ may be identical with the original individual $X_i(t)$ after crossover operation, resulting in invalid crossover. If invalid crossover occurs, a path with the same source node and destination node as $V_i(t)$ is randomly selected from population Pop as the individual $W(t)$, and the crossover individual $U_i(t)$ is obtained after crossover operation with the mutant individual $V_i(t)$. Thus, the probability of the results of mutation operation being retained to the next generation population can be improved. The schematic diagram of crossover operation is shown in Fig. 4. It is assumed that the individual $W(t)$ (Fig. 4 (a)) is cross operated with the mutation individual $V_i(t)$ (Fig. 3 (c)). Their first public node is $s_2^{l^*}$. All the link strings between $W(t)$ and $V_i(t)$ from the public node $s_2^{l^*}$ to the destination node s_d^e are exchanged to obtain the crossed individuals $U_i(t)$ as shown in Fig. 4 (b).



(a) Randomly selected individuals $W(t)$



(b) Cross the individual $U_i(t)$

Fig. 4. Crossover example

(4) Select operation

According to formula (9), the fitness values of the crossover individual $U_i(t)$ are calculated. If the fitness value of $U_i(t)$ is better than that of the current individual $X_i(t)$, otherwise, the crossed individual $U_i(t)$ in the t iteration replaces the current individual $X_i(t)$ and is retained to the next generation population; otherwise, the individual $X_i(t)$ is retained.

$$X_i(t+1)=\begin{cases} U_i(t), & if \ F(U_i(t)) > F(X_i(t)) \\ X_i(t), & otherwise \end{cases} \tag{9}$$

(5) If iteration conditions are met, exit the loop. Otherwise, go to step (2).

5. Experimental results and analysis

5.1 Experimental environment

To verify the effectiveness of DE-SDN algorithm proposed in this paper, the simulation experiment uses Floodlight as SDN controller. Mininet, a simulation platform, builds a k=4 Fat-Tree data center network, and uses Sflow technology to monitor network status [20]. The DE-SDN was compared with ECMP and GFF. The average bandwidth and maximum link utilization are used to measure the performance of the algorithm. Among them, the bisect bandwidth [21] refers to dividing a network evenly into two identical subnetworks, the total bandwidth of the data traffic that passes through all the links in the two subnetworks in a specified unit of time. The maximum link utilization is the value of the highest link utilization among all paths on the network during network transmission. In the same load case, the larger the pair bandwidth is, the larger the network throughput is. The lower the maximum link utilization is, the more evenly the network link utilization is, and no link is overused. In other words, the better the network load balancing performance is.

(1) Topology

Mininet platform is adopted to build a 4-yuan Fat-Tree network topology with Python programming [22], as shown in Fig. 5. All nodes are OpenFlow switches, and there are 20 in total [23]. Access layer switches connect 16 hosts, and the bandwidth of each link is set at 100Mb/s [24].

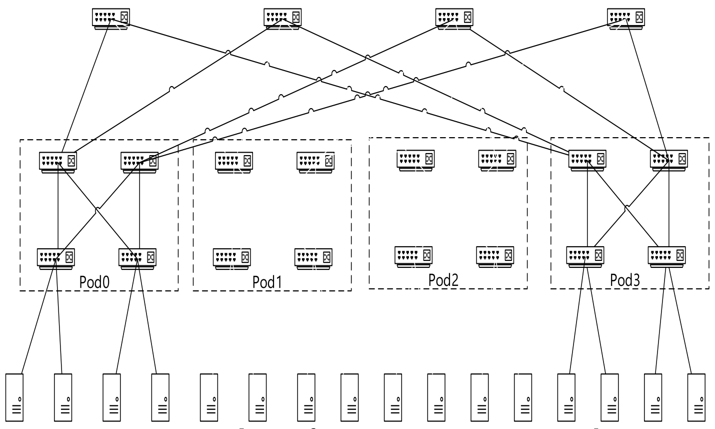


Fig. 5. Topology of a 4-yuan Fat-Tree network

(2) Communication mode

In literature [9], elephant flow is defined as data flow with bandwidth greater than 10% of link bandwidth, so data flow with bandwidth of 10Mb/s and above is identified as elephant flow in this paper. Currently, no publicly available data center network load data is available due to privacy and security issues. Therefore, this paper uses the communication mode used in reference [8, 11, 14, 15] to evaluate the elephant flow scheduling mechanism DE-SDN.

Iperf, a traffic generation tool, was used for secondary development. Data flows of three different communication modes were generated by extending the internal commands of Mininet. The three different communication modes are described below:

1) Random mode: The source and destination hosts are randomly selected on the network, and the traffic generation mode and traffic volume are randomly generated.

2) Stride(*i*) mode: The host numbered *x* transmits data to the host numbered (*x*+*i*) mod *n*, where *n* is the number of hosts in the network [25].

3) Staggered (*p1*, *p2*) mode: Each host with probability *p1* to belong to a host sends data access layer switches, with probability *p2* to belong to a host of pod to send data, and with probability 1-*p1*-*p2* to other pods within the host to send data.

According to data center network characteristics [26], the size of data flow in the experiment obeys exponential distribution, in which the parameter *r* of exponential function is 0.23. The time interval for generating each flow obeys

Poisson distribution, and the duration of each flow is 60 seconds. Data flows from the 20th to the 40th seconds are taken as effective experimental data.

(3) Algorithm parameter setting

According to the optimization objective of traffic scheduling, the main parameter Settings of DE-SDN algorithm in the simulation experiment are shown in Table 1.

Table 1

Main parameter values of DE-SDN algorithm

Parameters	The set value	Recommended values
Maximum iteration *MaxT*	50	[50,100]
The number of population *M*	50	[50,100]
Impact factor *a*	1	[1,10]
Impact factor *b*	10	[1,10]
The threshold value *H*	0.5	0.5
crossed factors *cr*	0.1	[0,1]

Since the optimization objective is to minimize the maximum link utilization, the influence factors of fitness function are set as *a*=1, *b*=10. If the crossover factor *cr* is large, it will usually accelerate convergence and obtain the local optimal solution, so *cr* =0.1.

5.2 Average bandwidth comparison

The data flow types in data center network are complex and the traffic is huge. The simulation experiment selects the above three communication modes to compare the performance of ECMP, GFF and DE-SDN traffic scheduling algorithms. After 20 groups of experiments for each traffic scheduling algorithm in each communication mode, the average value is taken to obtain the final results. The comparison of the average bisection bandwidth of the experimental results is shown in Fig. 4-6, where the horizontal axis represents the communication mode and the vertical axis represents the average bisection bandwidth of the distribution, in Mbps/s.

(1) Random mode

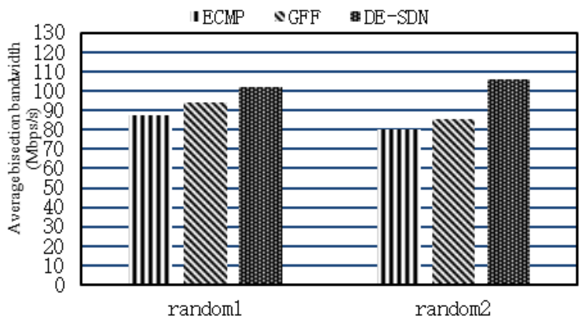


Fig. 6. Comparison of average bisection bandwidth in random mode

Two groups of experiments random1 and random2 are carried out in random mode. As can be seen from Fig. 6, the average bisection bandwidth of

GFF algorithm is higher than that of ECMP algorithm, and the average bisection bandwidth of DE-SDN algorithm is higher than that of ECMP algorithm and GFF algorithm. Compared with ECMP algorithm, DE-SDN algorithm improves the average bisection bandwidth by 16.74%~32.09%, and compared with GFF algorithm, the average bisection bandwidth increases by 7.95%~23.62%.

(2) Stride (i) mode

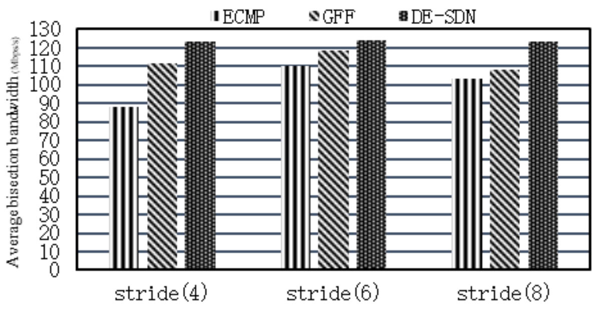


Fig. 7. Comparison of average bisection bandwidth in stride mode

In order to simulate the load imbalance of the network and facilitate the simulation of the traffic complexity of the real network, three groups of stride(4), stride(6) and stride(8) were compared in stride(*i*) interval mode with three communication modes: *i* =4, 6 and 8. Fig. 7 shows that in the three interval modes, the average bisection bandwidth of DE-SDN algorithm is higher than that of ECMP and GFF algorithm. In the Stride (6) mode, the average bisection bandwidth of DE-SDN algorithm is 12.69% higher than that of ECMP algorithm and 4.89% higher than that of GFF algorithm.

(3) Staggered (p1, p2) mode

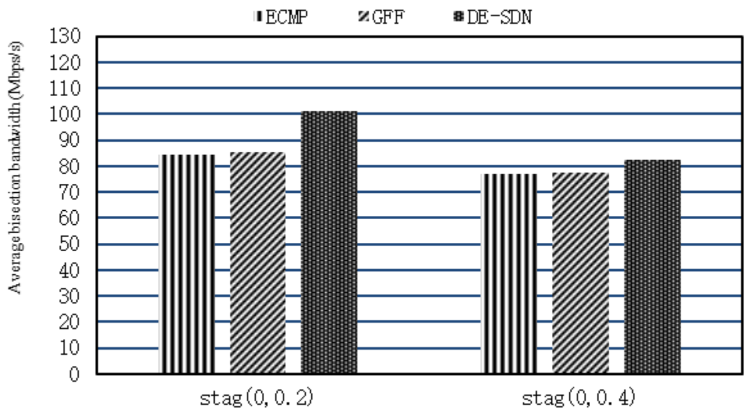


Fig. 8. Comparison of average bisection bandwidth in staggered mode

In staggered (*p1*, *p2)* modes, staggered (0, 0.2) and Staggered (0, 0.4) are selected for experiments. Staggered (0, 0.2) is used as an example, in which each host sends data streams to a host with the same Pod with a probability of 0.2. Sends data streams to other hosts in the remaining Pods with a probability of 0.8. As shown in Fig. 8, under these two interleaved modes, there is no significant difference between the average bisection bandwidth of ECMP algorithm and GFF algorithm, but the average bisection bandwidth of DE-SDN algorithm is higher than that of ECMP algorithm and GFF algorithm. Among them, the average bisection bandwidth of DE-SDN algorithm is 20% higher than that of the ECMP algorithm on staggered (0, 0.2) mode. On staggered (0, 0.4) modes, DE-SDN was 6.7% better than ECMP.

To sum up, in the three communication modes, DE-SDN algorithm has higher average pair bandwidth than ECMP algorithm and GFF algorithm. This is because the ECMP algorithm is a static hash scheduling algorithm, which evenly distributes data flows to different equivalent paths according to the hash value. Without considering the real-time state of the network, it cannot properly handle elephant flow scheduling, resulting in increased data flow conflicts and link congestion. Compared with THE ECMP algorithm, the GFF algorithm selects the first qualified path for rerouting elephant flows on congested roads according to the current network information status and the links through which data flows pass, thus improving the average bisection bandwidth of the pair allocation. However, in staggered mode, due to the increase of data flows between Pods, the number of conflicting flows also keeps increasing, and GFF still chooses the first path that meets the conditions. In this case, it is impossible to determine whether the path is the global optimal path, and network congestion may occur again, which cannot effectively alleviate the problem of load imbalance. The DE-SDN algorithm proposed in this paper firstly performs scheduling of ECMP algorithm. When real-time monitoring finds link congestion caused by elephant flow, a global optimal path is selected based on the current network link status to reschedule elephant flow. Therefore, compared with the other two algorithms, the average bisection bandwidth is better.

5.3 Maximum link utilization comparison

The maximum link utilization is also an important index to evaluate whether the traffic scheduling algorithm can better achieve network load balancing. It can reflect the network link usage. If the network load is unbalanced, the links with high network load will become congested, and the links with low network load will become redundant because the links are not fully utilized, resulting in the decrease of the bisection width.

To simulate real network load imbalance, staggered (0, 0.2) communication modes were selected to compare the maximum link utilization of ECMP, GFF and DE-SDN. 20 experiments were carried out for each algorithm, and the transmission duration of each data stream was 60 seconds. Finally, the cumulative distribution function values of all the maximum link utilization values were calculated. The frequency of each interval was 0.1, and the broken line graph as shown in Fig. 9 was obtained. In the figure, the abscissa represents the maximum link utilization, the ordinate is the cumulative distribution function value, and the three curves are the cumulative distribution function curves corresponding to ECMP, GFF and DE-SDN algorithms. The higher the position of the cumulative distribution function curve, the better the effect of reducing the maximum link utilization is. As can be seen from the figure, the maximum link utilization of ECMP algorithm is mostly concentrated in 70%~85%, and that of GFF algorithm is 60%~80%, while the maximum link utilization of DE-SDN algorithm is about 10% lower than that of ECMP and GFF algorithm, and most of them are concentrated in 60%~75%.

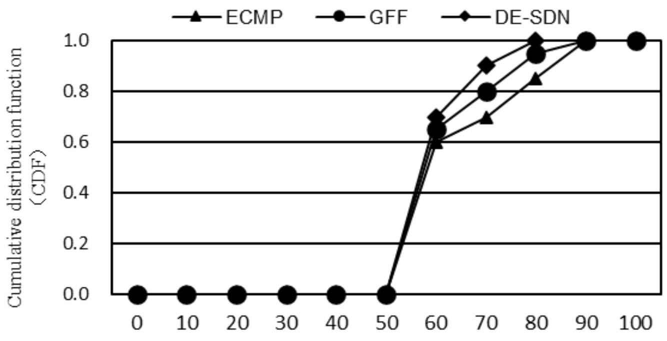


Fig. 9. Comparison of maximum link utilization in staggered (0, 0.2) mode

6. Conclusions

Aiming at the problems of link congestion and load imbalance existing in traditional data center network traffic scheduling algorithm, this paper proposes a traffic scheduling mechanism of SDN data center network DE-SDN based on differential evolution algorithm, considering the great impact of elephant flow on the network. According to the actual network state and link integrity rules, the crossover operation and mutation operation of differential evolution algorithm are improved. Through simulation experiments, it is further verified that the proposed DE-SDN algorithm not only improves the average matching bandwidth, but also reduces the maximum link utilization, and provides a guarantee for network load balancing by reasonably scheduling elephant flow compared with the classical ECMP and GFF algorithms. Although this experiment has achieved good experimental results, there are still many problems. This experiment uses a single controller, easy to cause a single point of failure in the network. Therefore, the next step is to realize the multi-controller scheduling mechanism of SDN and expand the experimental scale, and also refine the traffic scheduling scheme by combining different types of controllers.

REFERENCES

[1]. *W. X. LI, H. QI and R. H. XU,* Data Center Network Flow Scheduling Progress and Trends in Chinese Journal of Computers, vol. 43, no. 4, 2020, pp. 600-617.
[2]. *Y. P. CAI, C. P. WANG,* Software defined data center network with hybrid routing in Journal on Communications, vol. 37, no. 4, 2016, pp. 44-52.
[3]. *D. Q. Li,X. Wang,Y. N.Jin and H. X.Liu,* Research on QoS routing method based on NSGAII in SDN in Journal of Physics: Conference Series, vol. 1656, 2020.
[4]. *Z. Qingyun, C. Ming, Z. Guangsong,* Research on SDN based on OpenFlow in J. Softw, vol. 24, no. 5, 2013, pp.1078-1097.
[5]. *Hopps C.*, Analysis of an Equal-Cost Multi-Path Algorithm in RFC 2992, IETF, RFC 2000
[6]. *D. Gang, Z. H. Gong, H. Wang,* Characteristics Research on Modern Data Center Network in Journal of Computer Research and Development, 2014, vol. 51, no. 02, pp. 395-407.
[7]. *T. Benson, A. Akella, D. A. Maltz,* Network traffic characteristics of data centers in the wild in Proc of the 10th ACM SIGCOMM Conference on Internet Measuremen,2010, pp. 267-280.
[8]. *M. Al-Fares*, S. Radhakrishnan, Raghavan B. et al. Hedera: dynamic flow scheduling for data center networks in Usenix Symposium on Networked Systems Design and Implementation, NSDI 2010, 2010, pp.281-296.
[9]. *Curtis A R, Kim W., Yalagandula P.*, Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection in Proc of IEEE INFOCOM., 2011, pp.1629-1637.
[10]. *S. Chakraborty, C. Chen, A* low-latency multipath routing without elephant flow detection for data centers in IEEE, International Conference on High Performance Switching and Routing. IEEE, 2016, pp.49-54.
[11]. *Z. P. LIU, S. S. REN, M. LI, X. P. WANG, X. F. LI,* Software defines dynamic traffic scheduling scheme for network data center in Journal of Jilin University (Engineering and Technology Edition) , vol. 51, no.03, 2021, pp.1040-104.
[12]. *S. X. ZHU, Y. F. LONG, G. L. SUN,* Software-defined Data Center Network Load Balancing Algorithm Based On Large Flows Scheduling in Computer Applications and Software, vol. 38, no.01, 2021, pp.27-32+75.
[13]. *D. Q. PENG, X. W. LAI, Y. L. LIU,* Multi-path Routing Algorithm for Fat-tree Data Center Network Based on SDN in Computer Engineering, vol. 44, no.04, 2018, pp.41-45+65.
[14]. *Z. H. LIN, W. GAO, C. M. WU,* Data Center Network Flow Scheduling Based on DPSO Algorithm in Acta Electronica Sinica, vol. 44, no.09, 2016, pp.2197-2202.
[15]. *H. H. LI, G. YANG, H. L. LU, X. L. FU, Z. J. SHEN,* Flow scheduling of elephant flows in SDN data center network based on ant colony algorithm in Application Research of Computers, 2019, pp.1-7.
[16]. *R. Storn and K. Price,* Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces in Journal of Global Optimization, vol. 11, no.04, 1997, pp. 341-359.
[17]. *Q. F. DING, X. Y. YIN,* Research survey of differential evolution algorithms in CAAI Transactions on Intelligent Systems, vol. 12, no.04, 2017, pp.431-442.
[18]. *Y. L. Gao,J. M. Liu, Y.H. Yao,* Multiobjective Differential Evolution Algorithm with Multiple Trial Vectors in Abstract and Applied Analysis, 2012.
[19]. *Y. J. Yen,* Finding the K Shortest Loopless Paths in a Network in Management Science, vol. 17, no.11,1971.
[20]. *Q. Tang, H. Zhang, J. Dong, L. M. Zhang, A. J. Peña,* Elephant Flow Detection Mechanism in SDN-Based Data Center Networks in Scientific Programming, 2020.
[21]. *K. Giotis , C. Argyropoulos , G. Androulidakis , et al.,* Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments in Computer Networks, vol. 62, no.05, 2014, pp.122-136.

[22]. *E. S. Jesus*, A new proposal to deal with congestion in InfiniBand-based fat-trees in Journal of Parallel and Distributed Computing, vol. 74, no.01, 2014, pp.1802-1819.

[23]. *H. H. Li, W. D. Li and X. L. Fu*, "Network Traffic Scheduling Algorithm For Energy-Efficient SDN Data Center Based On Multi-Layer Virtual Topology" in Computer Applications and Software, vol.38, no.04, 2021, pp. 124-131.

[24]. *L. Jin, Y. A. Shu*, "Research on load balancing of elephant flow based on SDN in data center network" in Application Research of Computers, vol. 36, no. 01, 2019, pp. 203-205.

[25]. *Li Honghui et al*, "An optimal and dynamic elephant flow scheduling for SDN-based data center networks" in Journal of Intelligent & Fuzzy Systems, vol.38, no.01, 2020, pp. 247-255

[26]. *L. Z. Tan, W. Su, P. Cheng, L.Y. Jiao, Z. Y. Gai*, Sonum: Software-Defined Synergetic Sampling Approach and Optimal Network Utilization Mechanism for Long Flow in a Data Center Network in Applied Sciences, vol.10, no.10, 2019.