doi:10.11835/j.issn.1000-582X.2021.201

采用 OpenFlow 交换机的服务器负载均衡策略

曾友雯,李双庆,邹东升
(重庆大学 计算机学院,重庆 400044)

摘要:云数据中心对服务器的海量并发访问十分普遍。传统网络架构难以全局控制流量转发,需要配置昂贵的负载均衡器应对这一应用场景。软件定义网络 SDN(software defined network)能够通过控制器全局掌控网络状态,并以交换机作为负载均衡器,从而降低部署成本。文中提出一种采用 OpenFlow 交换机的服务器负载均衡策略,通过多地址定向流表对服务请求进行分区映射,以活动连接数作为负载评估参数,通过蚁群算法求解最佳负载重定向方案。在负载迁移时,采用单地址定向流表来保证不同阶段流量的有序转发。实验结果显示,该策略能有效控制流表规模,并较传统均衡策略具有更优的性能。

关键词:SDN;OpenFlow 交换机;负载均衡;蚁群算法
中图分类号:TP393.0 **文献标志码**:A **文章编号**:1000-582X(2021)11-048-09

Server load balancing strategy using OpenFlow switch

ZENG Youwen, *LI Shuangqing*, *ZOU Dongsheng*
(College of Computer Science, Chongqing University, Chongqing 400044, P. R. China)

Abstract: Massive concurrent access to servers is very common in cloud data centers. It is difficult for traditional network architecture to control traffic forwarding globally, and expensive load balancers are needed to deal with this application scenario. The SDN(software defined network)can globally control the network status through the controller, and use the switch as a load balancer, thereby reducing deployment costs. A server load balancing strategy based on OpenFlow switches was proposed in this paper. Service requests were partitioned and mapped through multi-address directed flow table, and the number of active connections was used as a load evaluation parameter. The optimal load redirection scheme was found through ant colony algorithm. During load migration, single-address directed flow table was used to ensure orderly forwarding of traffic at different stages. Experimental results show that the proposed strategy can effectively control the size of flow table and has better performance than traditional equalization strategies.

Keywords: SDN; OpenFlow switch; load balancing; ant colony algorithm

收稿日期:2021-01-15 **网络出版日期**:2021-03-09
基金项目:国家自然科学基金资助项目(61309013)。
Supported by National Natural Science Foundation of China(61309013).
作者简介:曾友雯(1995—),女,硕士研究生,主要从事软件定义网络研究。
通讯作者:李双庆,男,副教授,(E-mail) sqlee@cqu.edu.cn。

随着云计算的兴起,云数据中心往往通过负载均衡手段协调服务器池中的服务器负载分配,以获得优化的响应性能[1-2]。近年来,软件定义网络 SDN(software defined network)在云数据中心广泛应用。SDN 将控制平面与数据平面解耦合,由 SDN 控制器感知网络状态[3-4]。在 SDN 网络中,控制器通过 OpenFlow 协议[5]动态变换交换机中的流表规则,实现对网络的动态控制。一些研究利用这一特点,通过动态变换用作负载均衡的交换机 LBS(load balancing switch)中的流表规则来实现服务器间的负载均衡。负载均衡的策略集中在控制器端,其优点是通过软件定义的方式动态部署不同的负载均衡策略,并且节省了传统负载均衡器方式的部署成本[6]。

近年来对 SDN 环境下的服务器负载均衡领域的研究主要集中在服务器均衡策略[7-8]、流表资源优化[9-11]等方面。Zhong 等[7]将用户请求定向到实时响应时间最短的服务器处理,使服务器保持负载均衡的状态。Handigol 等[8]提出 Plug-n-Server 流量负载平衡方案,将控制器模块化,通过控制器全局管理资源,使控制器高效有序地调度资源。Wang 等[9]使用通配符规则转发服务请求,将请求端按 IP 地址前缀划分为多个区块,根据负载动态分解或合并各区,以期占用 LBS 较少的流表项资源。Lin 等[10]提出通过给服务器分配优先级设置不同的通配符规则,适用于大规模数据中心,以更少的流表项达到更高效率。Mao 等[11]提出单流表和组流表结合的动态流表设计算法,负载均衡时调整更少流表项,减少负载均衡流表变换。

文中提出一种采用 OpenFlow 交换机的服务器负载均衡策略,使用分区通配符规则均衡转发服务请求,在 LBS 上预先部署多地址定向流表,SDN 控制器周期性采集 LBS 活跃连接数,据此计算出服务器负载状态。利用蚁群算法的信息正反馈和启发式搜索特性,在服务器负载偏差超过阈值时求解负载调度方案,迁移负载以达到负载均衡的目的。

1　基于通配符的流表规则

采用负载均衡策略的服务器集群对外以单一 IP 地址形成单一映像(single image)。作为负载分配器(load dispatcher)的 LBS 按照 SDN 控制器部署的流表规则转发请求到指定后端服务器。如果流表记录每个请求的转发规则,会产生流表溢出,影响转发效率,增大控制器处理开销[12-13]。

将服务请求按源 IP 地址前缀划分为多个区块,分别用通配符表示每个区块相同的地址前缀部分,将其映射到 OpenFlow 流表匹配字段中。在 LBS 转发服务请求时,由于多个请求匹配同一转发规则,因此称之为多地址定向流表 MDFT(multi-address directed flow table),如图 1 所示。MDFT 能有效解决流表过多的问题,主要用于流量转发和流量数据监控。

匹配字段	优先级	计数器	● ● ●
101*			
匹配字段	优先级	计数器	● ● ●
110*			
匹配字段	优先级	计数器	● ● ●
111*			
● ● ●			

图 1　多地址定向流表表项

Fig. 1　Multi-address directed flow entry

当需要平衡负载,将原区域的负载重新分配时,引入单地址定向流表 SDFT(single-address directed flow table)。在负载均衡调整期中同时存在调整前的 TCP 连接和调整启动后的新连接。SDFT 登记每条新请求的 TCP 连接,而调整前的流量则继续沿用原来的规则,从而保障 TCP 连接映射服务器的一致性。

在匹配规则上约定 SDFT 优先级高于 MDFT,当请求同时满足 SDFT 和 MDFT 匹配条件时,按照 SDFT 规则转发流量。多个 MDFT 可以将服务请求转发到同一服务器,但 MDFT 间 IP 通配符地址不能重叠,以保证服务请求转发到唯一服务器。MDFT 生存时间较长,在不需要调整流表项时,服务请求按照 MDFT 转发。SDFT 生存时间较短,仅在负载均衡调整期使用。

2 基于 OpenFlow 交换机的负载均衡策略

如图 2 所示，假设部署 x 台服务器 $S_i(i=1,2,\cdots,x)$，将请求端按源 IP 地址前缀初始划分为 m 个区块 $B_k(k=1,2,\cdots,m)$，每个区块分别映射一个 MDFT 表项，其 IP 地址前缀为 $I_{\text{pre\_k}}$，每个区块覆盖 n 个请求端源 IP 地址，SDN 控制器对应生成并下发 m 个流表项 $F_k(I_{\text{pre_k}},S_i,p)$到 LBS，其中 $k=1,2,\cdots,m$，p 为流表项的优先级。地址前缀满足 $I_{\text{pre_k}}$的服务请求通过 LBS 匹配流表项 $F_k(I_{\text{pre_k}},S_i,p)$后被转发到服务器 S_i。当不同服务器分配到的负载发生不均衡时，SDN 控制器需要及时感知并做出负载再分配以均衡服务器负载。

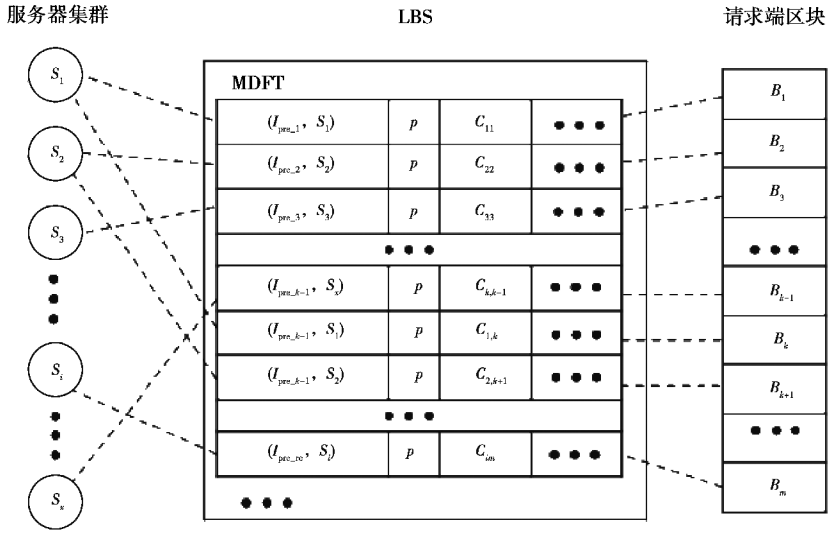

图 2 负载均衡交换机转发示意图

Fig. 2 Diagram of load balancing switch forwarding

2.1 服务器负载计算

在衡量服务器负载时，主要分为直接或间接方式获取服务器的负载状态。李艳冠等[14]和于天放等[15]通过控制器采用 SNMP 协议获取服务器 CPU 利用率、内存利用率等指标来计算服务器负载。文献[7]通过控制器发送 Ping 命令测试服务器实时响应时间，根据实时响应时间度量服务器负载。上述直接获取服务器负载状态的方法可以较准确地衡量服务器的负载情况，但该方式对服务器缺乏透明性，对服务器的响应方式有一定的要求，因此存在一定的局限性。另外，该方式需要控制器直接参与服务器数据采集，增加了控制器的开销。Boero 等[16]提出通过交换机服务请求的入队速率和处理速率衡量后端服务器负载不均衡程度，将交换机中入队速率大于处理速率的流量重路由，从而达到后端服务器负载均衡的目的。该方法采用对服务器透明的机制，虽不能计算出服务器实际负载，但可以有效衡量后端服务器负载不均衡程度，减少控制器资源消耗，不限制服务器的响应方式。

采用对服务器透明的负载计算方法，掌握服务器的负载状态，据此求解负载调度方案，迁移负载以达到负载均衡的目的。在给定服务器容量的情况下，服务器的活动连接数可以作为服务器负载状态的度量指标[17]，通过 SDN 控制器收集 LBS 各流表项中活跃连接数作为服务器负载度量依据，避免对服务器参与的依赖。假设 y 个流表项映射到服务器 S_i，每个流表项的活动连接数为 c_{il}，其中 $l=(1,2,\cdots,y)$，则服务器 S_i 总的活动连接数为 $C_i=\sum_{l=1}^{y}c_{il}$。服务器 S_i 通过该流表项转发的流量的负载 L_i 可以通过式(1)计算出

$$L_i=\theta * C_i, \tag{1}$$

式中，θ 为活跃连接数对服务器负载的影响因子。当服务请求到达 LBS，LBS 中匹配的流表项通过计数器对 TCP 的 SYN 位和 FIN 位计数，SDN 控制器周期性采集 LBS 流表项中的该计数值，从而计算出活动连接数。

考虑负载均衡出现抖动，采取 t 次采样数据计算平均数的方法来计算确定服务器平均负载

$$L_{\text{avg}}^{i}=\frac{\sum_{z=1}^{t}L_i[z]}{t}, \tag{2}$$

式中，$L_i[z]$为在第 z 时刻服务器的总负载，t 为采样次数。

通过方差表示服务器间负载失衡度 δ 为

$$\delta(t)=\frac{\sum_{i=1}^{x}(L_{\text{avg}}^{i}-L_{\text{avg}})^2}{x}, \tag{3}$$

式中,$\delta(t)$为 t 时刻服务器间负载失衡度。$\delta(t)$越大,表示服务器间的不平衡程度越大。SDN 控制器触发负载均衡策略的条件为

$$\delta(t)>\omega, \tag{4}$$

式中,ω 为负载失衡度阈值。

2.2　自适应动态负载均衡策略

文中提出一种自适应动态服务器负载均衡策略,利用 SDN 控制器向 LBS 部署 MDFT 和 SDFT 流表项重定向服务请求,SDN 控制器采用基于蚁群算法的负载重定向算法 ACO-LBSA(ACO-based load balance switching algorithm)来决策服务器间的负载迁移,以达到负载均衡。

2.2.1　负载重定向方案选择

蚁群算法[18-20]是一种模拟蚂蚁寻找食物过程中发现路径的智能优化算法,该算法具有分布计算、信息正反馈和启发式搜索的特点。在研究场景中,负载均衡策略需要实时监控全局负载变化,并自适应均衡负载分配,因此采用蚁群算法以达到较快的收敛速度和较高的求解准确度。将服务器负载情况作为蚁群算法的信息素,重定向活动连接数的倒数作为启发函数,服务器间负载失衡度作为路径长度,算法目标是求解过载服务器负载重定向最优方案。

假定在当前时刻服务器 S_i 过载。负载重定向算法过程描述如下:

1)初始化信息素浓度及每只蚂蚁。在云数据中心初始化阶段,考虑服务器之间的处理能力异构性,用服务器的负载处理能力对信息素 τ_{ki} 初始化:

$$\tau_{ki}=d_i, \tag{5}$$

式中,d_i 表示为服务器 S_i 处理负载的能力。创建包含 m 个 MDFT 表项、x 个服务器的蚂蚁对象,初始化服务器失衡度。蚂蚁随机选择路径(F_k,S_i),路径表示为将流表项 F_k 重定向到服务器 S_i。并将分配过的 MDFT 从搜索列表 g_h 中删除,加入到禁忌表。

2)选择重定向服务器。蚂蚁按照概率与轮盘赌算法选择服务器重定向 MDFT,在 t 时刻第 h 只蚂蚁选择将流表项 F_k 重定向到副本服务器 S_i 的概率

$$P_{ki}^{h}(t)=\begin{cases}\dfrac{[\tau_{ki}(t)]^{\alpha}[\eta_{ki}(t)]^{\beta}}{\sum_{r\in gh}[\tau_{kr}(t)]^{\alpha}[\eta_{kr}(t)]^{\beta}}, & i\in g_h\\ 0, & i\notin g_h\end{cases}, \tag{6}$$

式中:$\tau_{ki}(t)$表示 t 时刻将流表项 F_k 重定向到服务器 S_i 的路径上的信息素浓度;α 为信息素启发因子,表示蚂蚁在路径上留下的信息素的重要性;β 为期望启发因子,反映了启发函数的重要性。$\eta_{ki}(t)$为启发函数表示服务器 S_i 处理流表项 F_k 转发流量的可见度,本策略采用重定向活动连接数定义启发函数

$$\eta_{ki}(t)=\frac{1}{c_{ik}}。 \tag{7}$$

3)更新任务禁忌表、搜索列表和服务器负载情况。根据步骤 1 中的方法更新禁忌表、搜索列表和服务器负载情况。

4)每只蚂蚁形成一个局部最优解。当一次迭代结束后,每只蚂蚁按照式(8)选择最小失衡度的解为最优解,最优解加入到序列 List<(F_k,S_i)>中,其中 a 代表蚂蚁的数量,h 代表第 h 只蚂蚁

$$R=\min_{h=1}(\delta_i),h\in a。 \tag{8}$$

5)更新信息素。为了防止信息素堆积,对每只蚂蚁完成分配后,调整信息素:

$$\tau_{ki}(t+1)=(1-\rho)\times\tau_{ki}(t)+\sum_{h=1}^{m}\Delta\tau_{ki}^{h}(t), \tag{9}$$

$$\Delta\tau_{ki}^{h}(t)=\frac{W}{\delta_i}, \tag{10}$$

式中:ρ 为信息素挥发因子;$\Delta\tau_{ki}^{h}$为蚂蚁 h 完成一条路径后在(F_k,S_i)上释放的信息素浓度;W 为常数,表示蚂蚁循环一次释放的信息素总浓度。

6)重复进行迭代,直到达到最大迭代次数,计算出最优解,并给出最优负载重定向矩阵

$$\boldsymbol{E}=\begin{pmatrix} e_{11} & \cdots & e_{1x} \\ \vdots & & \vdots \\ e_{m1} & \cdots & e_{mx} \end{pmatrix}。 \tag{11}$$

若 e_{ki} 的值不为空,则表示将流表项 F_i 的负载重定向到服务器 S_j;若为空,则不重定向。

2.2.2 基于蚁群算法的负载重定向算法

根据前文求解到的最优负载重定向矩阵 $\boldsymbol{E}$,SDN 控制器向 LBS 下发流表项重定向服务请求,实现负载均衡。基本思路为:SDN 控制器下发优先级为 p_1($p_1>p$)的流表项 $F_{\text{new}}(I_{\text{pre_k}},S_i,p_1)$和转发到新服务器 S_j 流表项 $F_{\text{new}}(I_{\text{pre\_k}},S_j,p)$,当服务请求到达 $F_{\text{new}}(I_{\text{pre_k}},S_i,p_1)$时,流表项检查 TCP 标识 SYN 位。若 SYN=1,视为新的 TCP 流,由 SDN 控制器下发优先级为 p_2($p_2>p_1$)的源 IP 地址为 I_{kj} 的 SDFT 表项 $f_{\text{new}}(I_{kj},S_j,p_2)$定向到新的服务器 S_j,并将流表项加入 SDFT 表项集合 f_j,将新的 TCP 请求重定向到新的服务器处理;若为原来持续的流,由 SDN 控制器下发流表项 $f_{\text{new}}(I_{ki},S_i,p_2)$,设置空闲超时时间 idle_time,从而保障 TCP 连接映射服务器的一致性,并将流表项加入集合 f_i。通常情况下[9],若在 60 s 内无数据传输,视为连接关闭,因此实验中设置 idle_time 为 60 s。超过 idle_time 后,删除 f_j 和 $F_{\text{new}}(I_{\text{pre\_k}},S_i,p_1)$,查看 f_i,若为空则删除集合,实现无差错的负载重定向。该算法伪代码如表 1 所示。

表 1 ACO-LBSA 算法

Table 1 Algorithm ACO-LBSA

ACO-LBSA

输入:云数据中心网络拓扑 G、定向到服务器 S_i 的 MDFT 表项 $F(I_{\text{pre\_k}},S_i,p)$

输出:重定向到服务器 S_j 的 MDFT 表项 $F_{\text{new}}(I_{\text{pre\_k}},S_j,p)$

```
while δ(t)>ω do
Select the S_i with the max(load);
L_i=θ*C_i;
E = ACO(F(I_pre_k,S_i,p));//利用蚁群算法求解服务器 S_i 负载调度方案
F.add(F_new(I_pre_k,S_i, p_1)), p_1> p;//下发流表项 F_new(I_pre_k,S_i,p)到 LBS
F.add(F_new(I_pre_k,S_j,p));
Set time automatically increases;//设置 time,time 自动增加
While match F(I_pre_k,S_i, p_1) and time < idle_time do
  If SYN == 1 do
    F.add(f_new(I_kj,S_j,p_2)), p_2> p_1;
    f_j.add(f_new(I_kj,S_j,p_2));//添加流表项 f_new(I_kj,S_j,p_2)到集合 f_j 中
  Else
    F.add(f_new(I_ki,S_i,p_2));
    idle_time = 60;//设置空闲超时时间为 60 s
    Record Nidle_time;//记录目前连接空闲时间
    f_i.add(f_new(I_ki,S_i,p_2));//添加流表项 f_new(I_ki,S_i,p_2)到集合 f_i 中
    If Nidle_time > idle_time do // 目前空闲时间超过空闲超时时间
      delete f_new(I_ki,S_i,p_2);
      f_i.remove(f_new(I_ki,S_i,p_2));
    End if
  End if
End while
delete F_new(I_pre_k,S_i, p_1);
delete f_j;
If  f_i is ϕ do
  delete f_i;
End if
End while
Return F_new(I_pre_k,S_j,p);
```

3　实验结果与数据分析

3.1　实验环境

研究在 Ubuntu 环境下使用 Mininet 搭建 SDN 网络仿真实验环境，选择 Ryu 作为 SDN 控制器，选择 OpenVSwitch 作为负载均衡交换机，构建实验网络，该网络拓扑结构如图 3 所示。

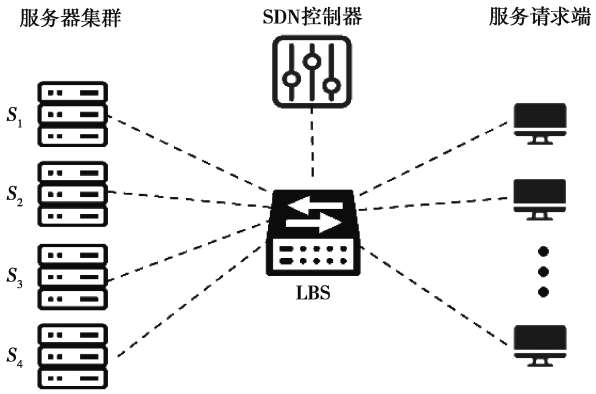

图 3　实验网络拓扑

Fig. 3　Experimental network topology

实验为服务器集群配置 4 台服务器，服务请求端选择整个 IP 地址空间的一个子集，将其划为 24 个初始区块，每个服务器初始按平均方式各分配 6 个区块。

3.2　实验分析

以云数据中心为目标，模拟 4 种场景下分别使用 Random 算法、Round Robin 算法和 ACO-LBSA 算法进行均衡负载的过程。4 种场景分别为：①低负载运行，服务器负载均在 10%～20%；②负载失衡度较高，不同区块的负载相差可达 10 倍；③服务请求突增，某些区块的负载在某个时刻陡增一倍；④高负载运行，服务器负载均在 80%～90%。如图 4～图 7 所示，为 4 种场景下发生一次负载均衡过程前后总共 180 s 内服务器平均响应时间，其中每隔 5 s 计算一次平均响应时间。

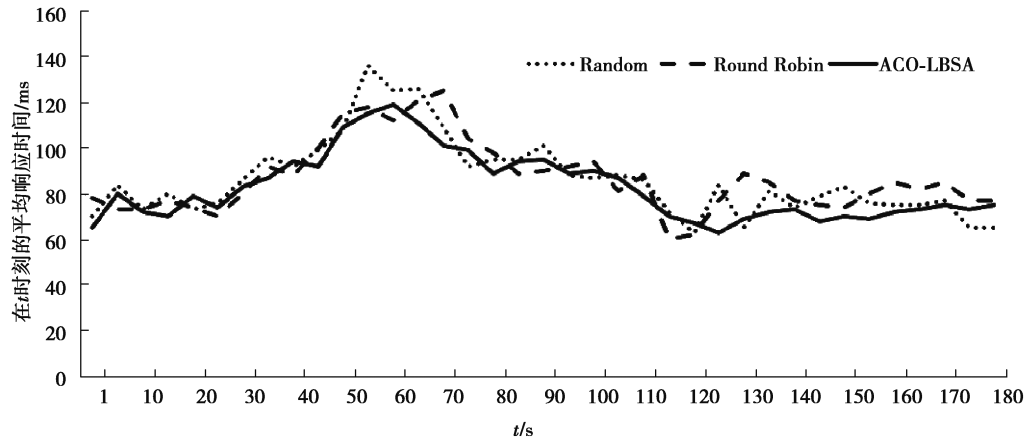

图 4　场景(1)下服务器平均响应时间变化趋势

Fig. 4　Trend of average server response time under scenario (1)

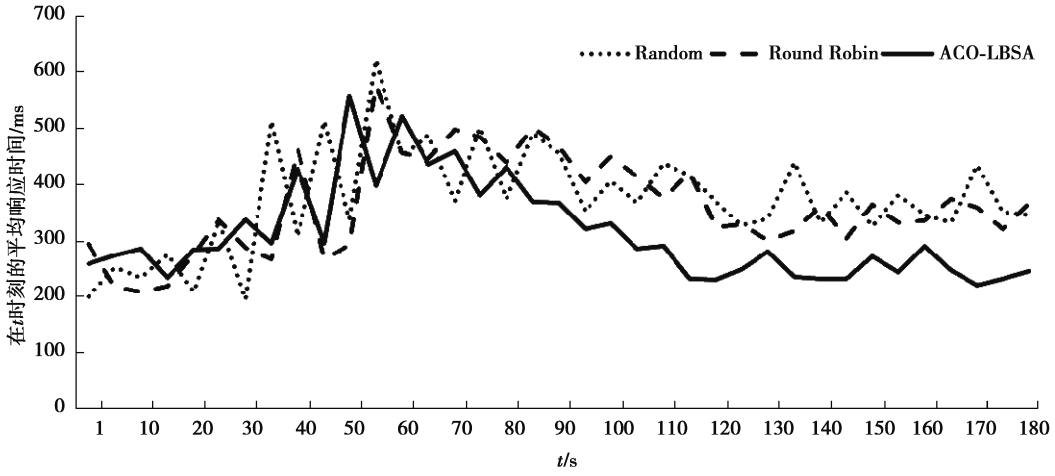

图5 场景(2)下服务器平均响应时间变化趋势

Fig. 5 Trend of average server response time under scenario (2)

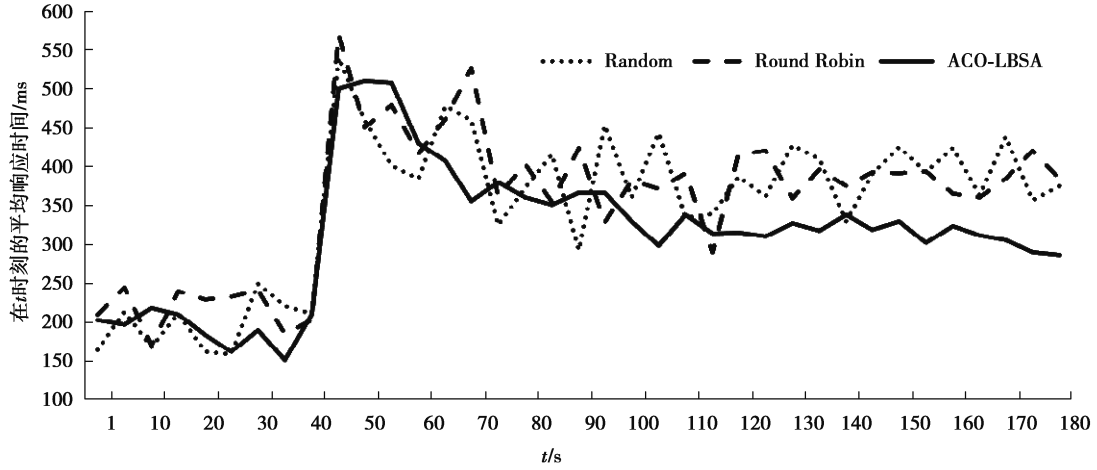

图6 场景(3)下服务器平均响应时间变化趋势

Fig. 6 Trend of average server response time under scenario (3)

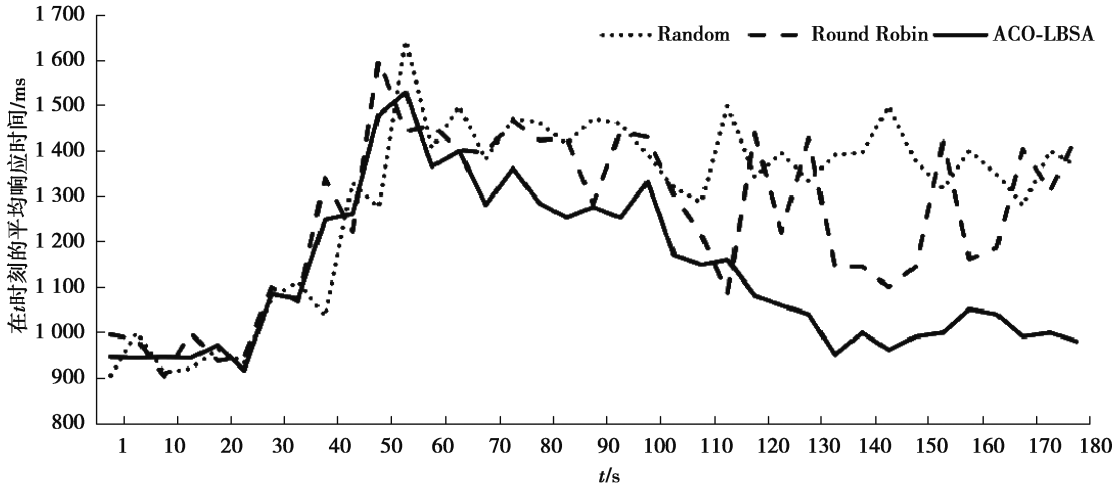

图7 场景(4)下服务器平均响应时间变化趋势

Fig. 7 Trend of average server response time under scenario (4)

如表 2 所示，为分别采用 ACO-LBSA 算法与 Random 算法、Round Robin 算法在 4 种场景下服务器响应性能对比。

表 2　服务器响应性能对比

Table 2　Comparison of server response performance

场景	Random 算法		Round Robin 算法		ACO-LBSA 算法	
	总的平均响应时间/ms	响应时间方差	总的平均响应时间/ms	响应时间方差	总的平均响应时间/ms	响应时间方差
场景 1	86.45	17.32	86.95	15.75	82.75	14.82
场景 2	372.68	95.51	362.86	89.18	312.03	86.49
场景 3	348.62	101.02	357.48	95.21	313.94	89.83
场景 4	1 297.78	197.43	1 254.29	187.18	1 128.73	171.29

从图 4～图 7 和表 2 中可以看出，在场景(1)下，采用 3 种算法的服务器平均响应时间相差不大，但采用 ACO-LBSA 算法的服务器平均响应时间略有优势，且服务器平均响应时间波动更小；在场景(2)下，采用 ACO-LBSA 算法的服务器平均响应时间明显优于其他 2 种算法，且服务器平均响应时间波动更小；在场景(3)下，采用 ACO-LBSA 算法的服务器平均响应时间明显优于其他 2 种算法，且服务器平均响应时间的波动明显降低；在场景(4)下，采用 ACO-LBSA 算法的服务器平均响应时间优化较场景(1)～场景(3)下程度更高，服务器平均响应时间波动较其他算法明显降低。

根据以上实验可以看出，文中提出的 ACO-LBSA 算法较 Random 算法和 Round Robin 算法在服务器响应性能方面有所改善，在云数据中心处于负载失衡度较高、服务请求突增和高负载运行等典型场景下，较其他 2 种算法对服务器响应性能改善更为明显。

4　结束语

提出了一种采用 OpenFlow 交换机的负载均衡策略。通过通配符规则缩减流表项数量，分别采用 MDFT 和 SDFT 用于均衡的负载分配和负载调整期的负载过渡，以活跃连接数度量服务器负载，从而使负载均衡对服务器保持透明。当服务器负载失衡度超过阈值，SDN 控制器通过蚁群算法求解出最优的负载调度方案，并向 LBS 下发 MDFT 流表项和 SDFT 流表项重定向服务请求，以达到负载均衡。通过仿真实验显示文中的策略较随机和轮转策略性能更优。后续研究工作将在此基础上讨论引入区分服务的负载均衡策略。

参考文献：

[1] 沈耿彪，李清，江勇，等. 数据中心网络负载均衡问题研究[J]. 软件学报，2020,31(7)：2221-2244.
Shen G B，Li Q，Jiang Y，et al. Research on load balancing in data center networks[J]. Journal of Software，2020,31(7)：2221-2244.(in Chinese)

[2] 王晶，何利力. 基于虚拟机动态迁移的负载均衡策略[J]. 计算机系统应用，2020,29(5)：167-174.
Wang J，He L L. Load balancing strategy based on dynamic migration of virtual machine[J]. Computer Systems & Applications，2020,29(5)：167-174.(in Chinese)

[3] 周桐庆，蔡志平，夏竟，等. 基于软件定义网络的流量工程[J]. 软件学报，2016,27(2)：394-417.
Zhou T Q，Cai Z P，Xia J，et al. Traffic engineering for software defined networks[J]. Journal of Software，2016,27(2)：394-417.(in Chinese)

[4] 姜文醍，谷宇，任丹妮，等. SDN 中基于流特征的 DDoS 攻击与闪拥事件检测[J]. 重庆邮电大学学报(自然科学版)，2019,31(3)：420-426.
Jiang W T，Gu Y，Ren D N，et al. DDoS attacks and flash crowds detection based on flow characteristics in SDN [J]. Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition)，2019,31(3)：420-

426.(in Chinese)

[5] Hamdan M, Hassan E, Abdelaziz A, et al. A comprehensive survey of load balancing techniques in software-defined network[J]. Journal of Network and Computer Applications, 2021, 174.

[6] Chen W, Shang Z, Tian X, et al. Dynamic server cluster load balancing in virtualization environment with openflow[J]. International Journal of Distributed Sensor Networks, 2015, 11(7): 8.

[7] Zhong H, Fang Y M, Cui J. LBBSRT: an efficient SDN load balancing scheme based on server response time[J]. Future Generation Computer Systems, 2017, 68: 183-190.

[8] Handigol N, Seetharaman S, Flajslik M, et al. Plug-n-Server: load-balacing web traffic using OpenFlow[C]. ACM Sigcomm Demo. 2009: 268-270.

[9] Wang R, Butnariu D, Rexford J. OpenFlow-based server load balancing gone wild[C]. Usenix Conference on Hot Topics in Management of Internet Cloud and Enterprise Networks and Services. USENIX Association. 2011: 12-12.

[10] Lin T L, Kuo C H, Chang H Y, et al. A parameterized wildcard method based on SDN for server load. balancing[C]// 2016 International Conference on Networking and Network Applications. July 23-25, 2016, Hakodate, Japan. IEEE, 2016: 383-386.

[11] Mao Q L, Shen W K. A load balancing method based on SDN[C]// 2015 Seventh International Conference on Measuring Technology and Mechatronics Automation. June 13-14, 2015, Nanchang, China. IEEE, 2015: 18-21.

[12] 李龙, 付斌章, 陈明宇, 等. Nimble: 一种适用于 OpenFlow 网络的快速流调度策略[J]. 计算机学报, 2015,38(5): 1056-1068.
Li L, Fu B Z, Chen M Y, et al. Nimble: a fast flow scheduling strategy for OpenFlow networks[J]. Chinese Journal of Computers, 2015,38(5): 1056-1068.(in Chinese)

[13] Jiang W R, Prasanna V K. Scalable packet classification on FPGA[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2012, 20(9): 1668-1680.

[14] 李艳冠, 甄涛. SDN 中基于负载均衡机制的网管技术设计[J]. 计算机与网络, 2017,43(1): 65-67,70.
Li Y G, Zhen T. Design on network management technology in SDN based on load balance mechanism[J]. Computer & Network, 2017,43(1): 65-67,70.(in Chinese)

[15] 于天放, 芮兰兰, 邱雪松. 基于软件定义网络的服务器集群负载均衡技术研究[J]. 电子与信息学报, 2018,40(12): 3028-3035.
Yu T F, Rui L L, Qiu X S. Research on SDN-based load balancing technology of server cluster[J]. Journal of Electronics & Information Technology, 2018,40(12): 3028-3035.(in Chinese)

[16] Boero L, Cello M, Garibotto C, et al. BeaQoS: Load balancing and deadline management of queues in an OpenFlow SDN switch[J]. Computer Networks, 2016, 106: 161-170.

[17] Bryhni H , Klovning E . A comparison of load balancing techniques for scalable Web servers[J]. IEEE Network, 2000(4):58-64.

[18] 张家波, 袁凯, 吴昌玉. 一种基于链路质量的蚁群优化 VANET 路由算法[J]. 重庆邮电大学学报(自然科学版), 2020, 32(2): 185-191.
Zhang J B, Yuan K, Wu C Y. An ant colony optimization routing algorithm based on link quality for VANET[J]. Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition), 2020,32(2): 185-191.(in Chinese)

[19] Guan B X, Zhao Y H, Li Y. An improved ant colony optimization with an automatic updating mechanism for constraint satisfaction problems[J]. Expert Systems with Applications, 2021, 164: 114021.

[20] 王春娟. 基于蚁群算法的物联网链路负载均衡控制研究[J]. 信息技术, 2020,44(6): 121-124,129.
Wang C J. Load balancing control of IoT link based on ant colony algorithm[J]. Information Technology, 2020,44(6): 121-124,129.(in Chinese)

(编辑 詹燕平)