doi:10.3772/j.issn.1002-0470.2020.06.005

一种高效的数据中心流表与链路联合均衡算法①

付琼霄② 孙恩昌③ 王倩雯 李 萌 张延华

(北京工业大学信息学部信息与通信工程学院 北京 100124)

摘 要 针对基于软件定义网络(SDN)的数据中心中老鼠流带宽小、持续时间短且网络占比高、容易导致流表负载不均衡,而大象流容易引起链路负载不均衡的特点,本文提出一种高效的数据中心网络流表与链路联合均衡算法(JLBFTL)。该算法将新到来的流量全默认为老鼠流,提出路径流表评价指标评价并选取路由路径,实现流表负载均衡;周期性监测网络中的流量,对新监测到的大象流,提出路径链路评价指标评价并选取路径,实现链路负载均衡。当网络中有大量突发流量时,可能导致部分链路负载过重,造成链路负载不均衡,此时,本文提出通过选择合适的大象流,利用备份路径和组表进行有效分流实现链路快速均衡。仿真结果表明,本文提出的 JLBFTL 算法与 SRL + FlowFit、L2RM 算法相比,在丢包率、带宽利用率和吞吐量方面均有不同程度的改善,提高了网络性能。

关键词 软件定义网络(SDN);数据中心网络;流表负载均衡;链路负载均衡

0 引 言

随着云计算、大数据等技术的兴起,数据中心网络规模不断扩大[1]。具有多条并行路径的 Fat-tree[2]、VL2[3]、hypercube[4]等拓扑结构为数据中心网络提供了高带宽、高拓展性以及良好的容错能力,从而被广泛应用。但受限于传统网络管理部署困难等问题,网络性能无法得到保障[5]。软件定义网络(software defined network, SDN)的出现为解决以上问题提供了新思路,并越来越多地被部署到数据中心网络中。其借助集中控制的优势,实时获取网络全局视图和网络状态,为网络管理与流量控制提供了有利条件。OpenFlow 作为一种标准南向接口协议被广泛部署于 SDN 交换机中。然而,由于芯片的功耗和成本较高,大多数与 OpenFlow 协议兼容的交换机的流表容量有限。当网络中流量爆发时,容易引起流表溢出,网络丢包率升高[6]。

为了在数据中心网络中实现有效的流量管理,流量通常被分为两大类:大象流(elephant-flows)和老鼠流(mice-flows)[7]。大象流携带数据量多,持续时间长;老鼠流携带数据量少,持续时间短。特别地,在所有流中,大象流的占比不到 10%,但却承载了全网 80% 的带宽[8]。由此可知,大象流容易引发链路负载不均衡,导致链路拥塞。而老鼠流容易引发流表负载不均衡,导致流表溢出。

目前,基于 SDN 的数据中心网络负载均衡策略大多只考虑了链路负载均衡。L2RM[9]通过为网络中到来的流量提供备份路径来分担主路径流量,均衡网络链路负载。LABERIO[10]通过设置负载均衡阈值来判断网络链路负载均衡情况,根据链路状态计算网络负载均衡度,当负载均衡度超过阈值时,选择负载最高链路上的最大流进行调度。Sehery 等人[11]提出了 SRL 和 FlowFit 2 个链路负载均衡算法,SRL 作为路由初始化算法,随机选择 2 条等价最

① 国家自然科学基金(61671029,61601330,61571021),国家留学基金委高等学校青年骨干教师出国研修(201806545031),先进信息网络北京实验室(PXM2019_014204_500029),中国博士后科学基金面上项目(2018M640032)和北京市教委科技计划(KM201610005004,KM201610005007)资助项目。

② 女,1995 年生,硕士生;研究方向:软件定义网络;E-mail:441617658@qq.com

③ 通信作者,E-mail:ecsun@bjut.edu.cn

(收稿日期:2019-06-13)

短路径,其中负载最小的路径作为初始路径;当某条链路出现拥塞时,FlowFit 依次选择该链路上对负载影响最大的流进行调度,直到链路的负载小于拥塞阈值。以上策略只考虑到了网络中的链路情况,忽略了 SDN 中流表容量有限的问题,容易引发流表溢出。并且,后 2 种方案当网络链路负载不均衡时选择最大流进行调度,容易造成调度路径负载过高甚至发生拥塞,不利于网络的链路负载均衡。

对于上述算法中的流表溢出问题,DIFF 算法[12]提出了流表负载均衡的概念。路由初始化时执行流表负载均衡策略以缓解流表溢出;在最大最小公平带宽分配原则下,对大象流进行重路由,以实现最大吞吐量。在这里,大象流大小被视为一致,并且流量也被假设为不可分割的。在大象流大小差异大、网络负载较重的环境下,该方案难以达到好的负载均衡效果。

针对以上算法的缺点,本文提出一种高效的数据中心网络流表与链路联合均衡算法(joint load balancing algorithm for flow tables and links, JLBFTL)。结合数据中心网络流量特点,将新到达网络中的流量视为老鼠流进行流表负载均衡,并把新流中监测到的大象流调度到带宽资源更多的路径上。同时当网络链路负载不均衡时,选择合适的大象流,利用备份路径和组表实现快速有效的分流来使其恢复均衡。仿真结果表明本文算法能够有效提升网络性能。

1　基于 SDN 的数据中心网络架构

本文选取 Fat-tree 拓扑作为数据中心网络拓扑,由 SDN 控制器对网络中交换机进行集中控制,如图 1 所示[13]。Fat-tree 拓扑分为 3 层,自上而下分别为核心层、汇聚层和边缘层。对于参数为 k 的 Fat-tree 拓扑,其包含 k 个 Pod,每个 Pod 中汇聚层和边缘层交换机数均为 $k/2$,核心交换机总数为 $(k/2)^2$,各边缘交换机连接 $k/2$ 个服务器。最重要的是,各源目的节点间拥有 $(k/2)^2$ 条等价路径,为数据中心提供了高带宽及良好的容错能力。基于此网络架构,本文提出一种高效的数据中心网络流表与链路联合均衡算法。利用 SDN 集中控制的优势,对网络中的老鼠流和大象流进行有效管理,实现网络的流表负载均衡与链路负载均衡,从而缓解流表溢出和链路拥塞。

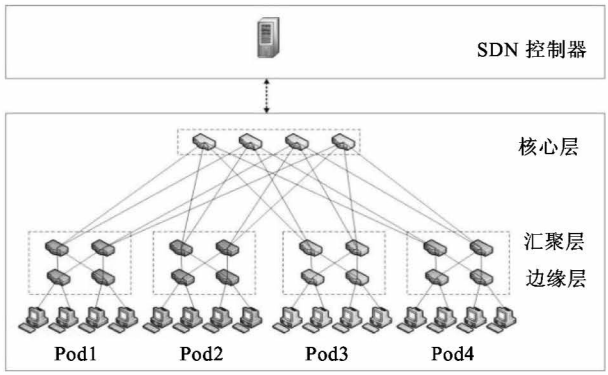

图 1　基于 SDN 的数据中心网络架构

2　流表与链路联合均衡算法

2.1　算法架构

本文提出的流表与链路联合均衡算法架构如图 2所示。主要包括备选路径集计算模块、路由初始化模块、大象流监测模块、大象流调度模块、分流模块以及流表下发模块。

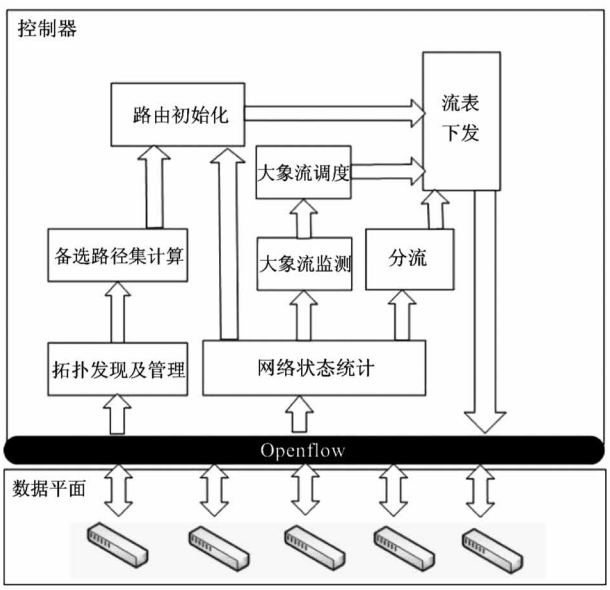

图 2　流表与链路联合均衡算法架构

2.1.1　备选路径集计算模块

网络初始化时,该模块针对 Fat-tree 拓扑特点,采用动态负载均衡(dynamic load balancing, DLB)中

的分层路由算法[14],得到各接入层交换机间的 k 条路径,并将所得路径存储到备选路径集中,为后续的路由算法所使用。本文仅考虑网络拓扑变化不频繁的情况,因此从预生成的备选路径集中选择路径能够显著减少控制器路由计算的计算量。

2.1.2 路由初始化模块

当网络中有新到来的流量,该模块从备选路径集中获取该流源目的节点间所有路径,并按照本文提出的路径流表评价指标对各条路径上的流表进行评分,选取评分最高的路径作为路由路径。

2.1.3 大象流监测模块

周期性监测源交换机传输流的总字节数和统计周期来估计流大小。流带宽大小可用式(1)[15]求出:

$$v = (b_{t+T} - b_t)/T \tag{1}$$

其中,b_t 为 t 时刻统计到的流传输字节数,b_{t+T} 为 $t+T$ 时刻的流传输字节数。

设置一个流量阈值 v_{th} 来区分大小流,当 $v > v_{th}$ 时,流量被判定为大象流,否则为老鼠流。

2.1.4 大象流调度模块

当监测到大象流,该模块对该流源目的节点间各条路径上的链路进行评分,选取评分最高的 2 条路径分别作为主路径和备份路径。其中,主路径用于大象流重路由,备份路径用于后期链路负载不均衡时为主路径分担部分流量。

2.1.5 分流模块

周期性监测网络链路负载均衡情况,当负载不均衡时,选取要进行分流的大象流,并计算主、备份路径上的流量分配比例,得到组表权重。

2.1.6 流表下发模块

该模块负责为路由路径上交换机下发流表及组表以实现流量转发。特别地,对于从大象流调度模块得到的主备份路径,为这 2 条路径上的所有交换机下发流表,并为源交换机下发组表,将大象流路由到主路径上。当分流模块被触发,调整源交换机组表权重以实现流量在主备份路径间的分配与传输。

2.2 算法描述

本文提出的流表与链路联合均衡算法主要分为流表负载均衡算法与链路负载均衡算法。流表负载均衡算法的主要目标是减少由网络中大量老鼠流导致的交换机流表溢出。同时,为了缓解链路拥塞,链路负载均衡算法为网络带宽的主要承载者大象流选择带宽资源最优的路径作为路由路径,并为其准备备选路径。在流量高峰期时,网络中大量的突发流量可能导致部分链路负载过重,造成链路负载不均衡,此时可通过选择合适的大象流,利用备份路径和组表进行有效分流实现链路快速均衡。

2.2.1 流表负载均衡算法

该算法运行于路由初始化模块中。由数据中心网络流量特点可知:老鼠流带宽需求小且持续时间短,对网络链路负载影响不大,但其数量庞大,占全网流量的绝大多数,会导致大量的流表下发,从而引发交换机流表溢出,导致网络丢包率升高。本文采用流表负载均衡策略解决以上问题。由于老鼠流在网络中的占比超过 90%,并且考虑到工程应用中的可行性,本研究将网络中到来的新流视为老鼠流[15],提出路径流表评分作为评价指标来评价源目的节点间的各条路径的流表质量,选择评分最高的路径作为路由路径,以实现流表均衡。

该指标综合考虑路径流表平均质量和路径瓶颈流表质量。由于本文中选择 Fat-tree 拓扑作为网络拓扑结构,一对源目的服务器间各条路径上交换机数一致,即流表数一致,所以路径流表平均质量可以由路径流表总质量表示。特别地,对于一个单独的流表而言,其剩余容量越大,负载越小,流表质量越好。然而,路径流表总质量不能简单地用路径流表剩余容量之和来评价,这样会忽略各流表间负载的差异。当各条路径上的流表剩余容量和均相同,每条路径的流表负载分布可能不同(例如,2 条路径上流表负载分布为[2,3,7,8]和[4,5,5,6])。但是这种不同不能从剩余容量之和中得到反映。此外,剩余容量之和高的路径也不一定比剩余容量之和低的路径更优。这是因为路径上流表的负载可能存在两极分化,路径上可能会有多个高负载流表,这样的路径是应该避开的。根据以上分析,本文提出路径流表负载总增量作为路径流表总质量的评价指标。

具体地,本文提出的流表负载增量函数 $T_i(x)$ 是一个非线性函数,用来增大流表间的差异,如式(2)

和式(3)所示。其代表当流量到达已存储流条目数为 x 的交换机 i 时,给该交换机流表带来的负担。其中,$f(x)$ 为流表负载评价函数。特别地,提出流表拥塞阈值,将流表状态分为 2 种:拥塞状态和非拥塞状态。在式(3)中,不同的函数被用来评价不同状态的流表。一个增长率更高的函数被用来评价拥塞状态的流表。通过这种方式,当流量到达一个部署了拥塞状态流表的交换机时,会产生很大的负载增量。

$$T_i(x) = f(x+1) - f(x) \tag{2}$$

其中,

$$f(x) = \begin{cases} \dfrac{1}{\alpha}x^{\alpha} = f_1(x) & x <= kC_{\max} \\ \dfrac{1}{\beta}(x-a)^{\beta} + b = f_2(x) & x > kC_{\max} \end{cases}$$

$$\beta > \alpha > 2 \tag{3}$$

a、b、α、β 取相应值使 $f(x)$ 满足连续、单调增,且为凸函数。$C_{\max}$ 为交换机流表容量,$kC_{\max}$ 为流表拥塞阈值。

考虑到路径上的瓶颈流表,综合路径流表负载总增量和路径最小流表剩余容量,提出路径流表评分计算公式,如式(4)所示:

$$score\_table = \frac{\min\{C_i\}}{\sum_i T_i(x)},\ i \in path \tag{4}$$

其中,C_i 为交换机 i 的流表剩余容量,$\min\{C_i\}$ 为路径 $path$ 上的最小流表剩余容量,$\sum_i T_i(x)$ 为路径的流表负载总增量。

流量将从源目的交换机间的多条备选路径中选择路径流表评分最大的路径进行传输。

流表负载均衡算法伪代码如算法 1 所示。

算法 1　Load Balancing for Flow Table

```
while new_flow do
  foreach path in paths do
    foreach sw in path do
      if table(sw) ≤ kC_max then
        load(sw) ← f1(table(sw))
      else
        load(sw) ← f2(table(sw))
      end if
      if table(sw) +1≤kC_max then
        load2(sw) ← f1(table(sw) + 1)
      else
        load2(sw) ← f2(table(sw) + 1)
      end if
      inc(sw) ← load2(sw)-load(sw)
      inc_table(path) ← inc_table(path) + inc(sw)
    end do
    min_free_table(path) ← min{C_max - table(sw)}
    score_table(path) ← min_free_table(path) / inc_table(path)
  end do
  path ← argmax{score_table(path) | path ∈ paths}
end while
```

其中,$table(sw)$ 为交换机 sw 中流表所存储的流条目数,$load(sw)$ 和 $load2(sw)$ 分别为流到达交换机 sw 前后该交换机中流表负载的评价值,$inc(sw)$ 为流表负载增量,$\min\_free_{table}(path)$ 为路径 $path$ 上流表的最小剩余容量,$score_{table}(path)$ 为路径流表评分。

2.2.2　链路负载均衡算法

链路负载均衡算法由大象流调度算法和分流算法组成,它们分别运行于大象流调度模块和分流模块中。

链路负载均衡的目标是缓解网络拥塞。由数据中心网络流量特点可知,大象流是网络链路带宽的主要占用者,网络链路负载均衡与其密切相关。但其数目只占流量总数的极小部分,对交换机流表负载影响不大。所以,链路负载均衡算法是针对大象流而言的。

(1)大象流调度算法

文献[9]为网络中所有流量提供主路径与备份路径。受该思想启发,并考虑到老鼠流占比大,为其成倍地下发流表会导致流表负担的急剧加重,且老鼠流数据量小、持续时间短,备份路径并不是必要的,因此本文只为大象流计算主备份路径。主路径用于大象流的重路由,以实现链路负载均衡;备份路径能够在后期网络链路状态不佳时,为主路径分担部分流量。此外,文献[9]并没有给出主路径和备份路径的选择方法,而本文给出了一种详细的选路方案。

主路径与备份路径的计算方法如下。类似于流表负载均衡算法中的思想,本文提出路径链路评分来评价源目的节点间的各条路径。其综合考虑了路径链路平均质量和瓶颈链路质量。因网络拓扑特点,路径链路平均质量可用路径链路总质量代替,这里路径链路负载总增量作为其评价指标。

具体地,本文提出链路负载增量函数 $L_{i,j}(x,v)$,同时考虑大象流大小以及链路负载情况,如式(5)所示。其代表将大小为 v 的大象流调度到已用带宽为 x 的链路上,给该链路带来的负担。

$$L_{i,j}(x,v) = g(x+v) - g(x) \tag{5}$$

其中,$g(x)$ 为链路负载评价函数,同 $f(x)$ 类似,唯一不同的是分界点的选取。在 $g(x)$ 中,分界点为链路拥塞阈值。当链路已用带宽超过该阈值,则判定该链路拥塞,选择增长速率更高的函数对该链路进行评价。因此,当大象流调度到拥塞链路上或大象流的调度导致调度链路的拥塞,都会产生很大的负载增量。

同时,考虑到路径的瓶颈链路,路径链路评分计算公式如式(6)所示:

$$score\_link = \frac{\min\{B_{i,j}\}}{\sum_{i,j} L_{i,j}(x,v)},\ i,j \in path \tag{6}$$

其中,$B_{i,j}$ 为交换机 i、j 间链路的剩余带宽,$\min\{B_{i,j}\}$ 为路径 $path$ 的最小链路剩余带宽,$\sum_{i,j} L_{i,j}(x,v)$ 为路径链路负载总增量。

大象流从源目的交换机间备选路径中选择路径链路评分最高和次高的路径分别作为主路径和备份路径。

大象流调度算法伪代码如算法 2 所示。

算法 2 Elephant _ flows Rerouting

```
1  while elephant _flow do
2    v ← speed(elephant _flow)
3    foreach path in paths do
4      foreach link in path do
5        if bw(link) ≤ kB_max then
6          load(link) ← g1(bw(link))
7        else
8          load(link) ← g2(bw(link))
9        end if
10       if bw(link) + v ≤ kB_max then
11         load2(link) ← g1(bw(link) + v)
12       else
13         load2(link) ← g2(bw(link) + v)
14       end if
15       inc(link) ← load2(link) - load(link)
16       inc_link(path) ← inc_link(path) + inc(link)
17     end do
18     min _free_bw(path) ← min{B_max - bw(link)}
19     score_link(path) ← min _free_bw(path) / inc_link(path)
20   end do
21   path_m ← argmax{score_link(path) | path ∈ paths}
22   paths2 ← remove path_m from paths
23   path_b ← argmax{score_link(path) | path ∈ paths2}
24 end while
```

其中,$bw(link)$ 为链路 $link$ 已用带宽,$load(link)$ 和 $load2(link)$ 分别为流量到达链路前后的链路负载评价值,$inc(link)$ 为链路负载增量,$\min\_free_{bw}(path)$ 为路径 $path$ 上链路的最小剩余容量,$score_{link}(path)$ 为路径链路评分。

(2)分流算法

当处于流量高峰期时,由于大量突发流量的到来,网络中某些链路可能会处于高负载状态,导致链路负载不均衡。针对该问题,本文提出分流算法。

该算法首先根据控制器周期性监测到的网络信息,计算链路负载均衡度。当判定网络链路负载不均衡时,选取要分裂的象流并计算组表权重,实现流量在主、备份路径间的分配。网络中的脉冲流(流带宽很大但持续时间短)可能对链路负载均衡度造成很大影响,导致不必要的分流,从而给网络带来能量消耗。因此,本文采用文献[9]中方案,当链路负载均衡度连续 m 个周期大于负载均衡阈值,则判断此时网络链路负载不均衡。

链路负载均衡度借鉴方差的概念,定义如式(7)[9]所示:

$$\delta = \frac{1}{N}\sum_{i,j\in path}(1 + |\ u_{i,j}(t) - u_{ave}(t)\ |)^2 \tag{7}$$

$$u_{i,j}(t) = \frac{B_{i,j}(t)}{C_{i,j}(t)} \tag{8}$$

其中,$B_{i,j}(t)$ 为 t 时刻交换机 i、j 间链路的已用带

宽，$C_{i,j}(t)$ 为链路最大带宽，$u_{i,j}(t)$ 为链路带宽利用率，$u_{ave}(t)$ 为全网链路平均带宽利用率，N 为网络中链路总数。由于 $|u_{i,j}(t) - u_{ave}(t)|$ 在 0 与 1 之间，添加 1 可使 δ 值变化范围更大，从而更易判断网络链路负载情况。

当监测到网络链路负载不均衡，选择全网链路利用率最高的链路上的最大的象流，并判断该流备份路径上是否存在拥塞链路。若不存在，计算组表权重，在主、备份路径间分配流量；若存在，从该大象流的多条备选路径中，选择除主路径之外路径最小剩余带宽最大的路径作为新备份路径并计算组表权重。

组表权重计算公式如下：

$$weight_{main} = \frac{B_{main} - v}{B_{back\_up} + B_{main} - v} \times 100 \quad (9)$$

$$weight_{back\_up} = \frac{B_{back\_up}}{B_{back\_up} + B_{main} - v} \times 100 \quad (10)$$

其中，B_{main}、B_{back_up} 分别为主路径、备份路径链路剩余带宽最大值，v 为选中分割的大象流流带宽。

分流算法伪代码如算法 3 所示。

算法 3 Flow Shunting

```
while link load imbalance do
  foreach link in links do
    utilization(link) ← port_state(link)
  end do
  link_max ← argmax{utilization(link) | link ∈ links}
  flow_set ← elephant_flow(link_max)
  foreach flow in flow_set do
    v(flow) ← flow_state(flow)
  end do
  flow_max ← argmax{v(flow) | flow ∈ flow_set}
  path ← path_backup(flow_max)
  if congestion links in path then
  calculate new path
    calculate weight
  else
    calculate weight
  end if
end while
```

其中，$utilization(link)$ 为链路 $link$ 的带宽利用率，$link_{max}$ 代表全网链路利用率最大的链路，$flow\_set$ 为该链路上的大象流集合，$flow_{max}$ 为该集合中的最大流。

3 仿真及结果分析

3.1 仿真场景描述

Mininet[16] 是一款可在 Linux 下运行的网络仿真器，能够方便地创建虚拟网络，本文的网络拓扑基于 Mininet 搭建。同时，选用开源的 SDN 控制器 Ryu[17] 作为本文的集中控制器，利用 iperf[18] 生成网络数据流。交换机选用支持 OpenFlow 协议的 Open vSwitch[19]。数据中心网络拓扑选用 $k = 4$ 的 Fat-tree 拓扑，即包含 20 台交换机和 16 台主机，如图 1 所示。实验中，链路带宽设置为 100 Mbps，交换机流表容量设置为 20。为了模拟数据中心网络流量，利用随机流量模型，各主机等概率地向网络中其他交换机发送数据流。其中，大象流所占比例小于老鼠流所占比例，且总和小于等于 1；大象流持续时间远远大于老鼠流持续时间。在这里假设大象流数量占 20%，老鼠流占 80%，且大象流持续时间设置为 60 s，老鼠流设置为 5 s。大象流与老鼠流的区分阈值设为链路带宽的 0.5%[10]，即 0.5 Mbps。

3.2 仿真结果及分析

本文将所提出的 JLBFTL 算法与 SRL + FlowFit、L2RM 算法进行对比，并从平均丢包率、平均带宽利用率及吞吐量 3 个方面比较了这 3 种算法的网络性能。由图 3、图 4 可得到表 1、表 2 中信息，从该信息中可以得知，相比于 L2RM 与 SRL + FlowFit，本文所提 JLBFTL 算法平均丢包率在整个区间上平均降低了 12.6% 和 11.6%，平均带宽利用率平均提升了 9.3% 和 14.7%。这是由于 SRL + FlowFit 只考虑了网络链路负载均衡，而忽视了交换机流表容量有限的问题，容易引发流表溢出，导致丢包；L2RM 虽通过动态调整流表空闲时间来缓解流表的溢出，但其随机选取初始化路由，容易使链路发生拥塞，造成丢包和包延迟。JLBFTL 同时考虑流表负载均衡与链路负载均衡，能够有效缓解流表溢出和链路拥塞，尤其保障了数目多但携带数据量少的老鼠流的正常传输，减少了丢包和包延迟，使得 JLBFTL 算法的平均

丢包率和平均带宽利用率得到改善。

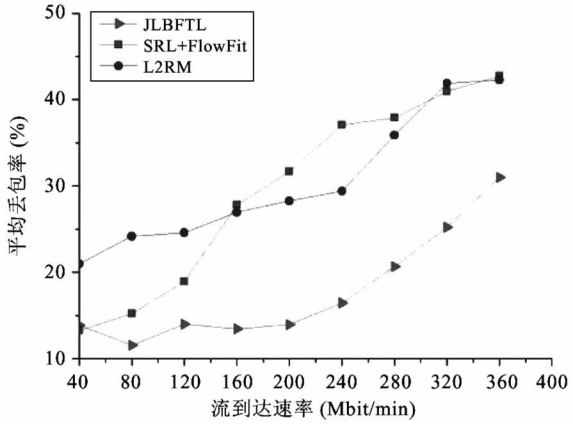

图 3　3 种算法平均丢包率比较

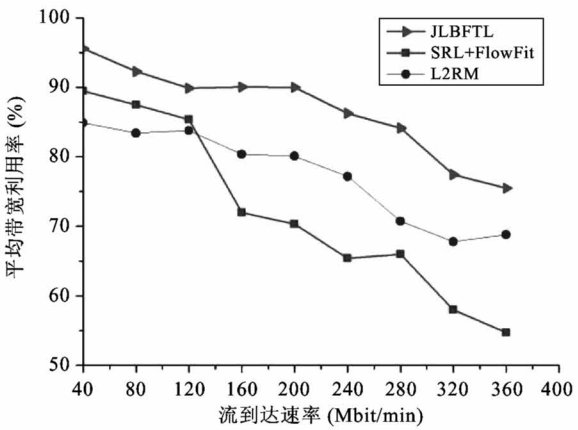

图 4　3 种算法平均带宽利用率比较

表 1　JLBFTL 相较于对比方案的平均丢包率下降率

流到达速率 (Mbit/min)	SRL + FlowFit(%)	L2RM(%)
40	-0.7	7.1
80	3.7	12.6
120	4.9	10.6
160	14.3	13.5
200	17.7	14.3
240	20.6	13.0
280	17.2	15.2
320	15.7	16.7
360	11.7	11.3
40 ~ 360	11.6	12.6

表 2　JLBFTL 相较于对比方案的平均带宽利用率增加率

流到达速率 (Mbit/min)	SRL + FlowFit(%)	L2RM(%)
40	6.1	10.7
80	4.8	8.9
120	4.5	6.1
160	18.1	9.7
200	19.7	9.9
240	20.9	9.1
280	18.2	13.4
320	19.5	9.7
360	20.8	6.7
40 ~ 360	14.7	9.3

由图 5 可知,在吞吐量方面,JLBFTL 相较于 L2RM 有一定的提升。虽然 L2RM 和 JLBFTL 在路由初始化时均未考虑链路状态,但 JLBFTL 对大象流进行了重路由以获得更高的链路带宽。因此,L2RM 在链路负载均衡方面效果不佳,无法保证大象流的传输,使其吞吐量低于 JLBFTL。JLBFTL 与 SRL + FlowFit 相比,当网络流量到达速率小于 280 Mbit/min 时,JLBFTL 吞吐量低于 SRL + FlowFit;随着流到达速率的增长,JLBFTL 吞吐量最终超过了 SRL + FlowFit。这是由于 SRL + FlowFit 在路由初始化时考虑了链路状态,而 JLBFTL 并未考虑,在该时期 JLBFTL 可能引发大象流的拥塞,导致大象流丢包,网络吞吐量降低。但随着网络流量的增多,网络负载的加重,SRL + FlowFit 的最大流调度方案可能导致新链路的拥塞。此时,JLBFTL算法的分流机制

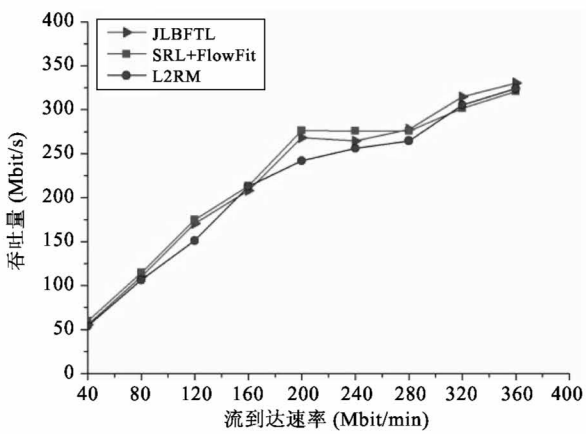

图 5　3 种算法吞吐量比较

使得大象流的传输得到保障,从而使吞吐量得到有效提升。

4 结 论

本文提出的 JLBFTL 算法,针对数据中心网络流量特征,将流表负载均衡和链路负载均衡相结合。考虑路径上流表负载总增量与最小流表剩余容量初始化路由,实现流表均衡。对于监测到的大象流,考虑路径上链路负载总增量与最小链路剩余带宽,将大象流调度到带宽资源更多的路径上,实现链路负载均衡。当流量高峰期时,网络中大量的突发流量可能导致部分链路负载过重,造成链路负载不均衡,此时可通过选择合适的大象流,利用备份路径和组表进行有效分流实现链路快速均衡。仿真结果表明,与 SRL + FlowFit、L2RM 算法相比,本算法在带宽利用率、丢包率、吞吐量方面都有一定的改善,缓解了网络拥塞和流表溢出,提升了网络性能。

由于本文算法在路由初始化时将所有流量都视为老鼠流进行流表均衡,可能会导致大象流的拥塞。这是未来工作中需要解决的问题。同时,基于数据中心网络大小流特点,可以进一步设计流表空闲超时时间以实现流表空间更合理的利用。

参考文献

[1] Neghabi A A, Navimipour N J, Hosseinzadeh M, et al. Load balancing mechanisms in the software defined networks: a systematic and comprehensive review of the literature[J]. *IEEE Access*, 2018, 6: 14159-14178

[2] Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture[J]. *ACM SIGCOMM Computer Communication Review*, 2008, 38(4): 63-74

[3] Greenberg A, Hamilton J R, Jain N, et al. VL2: a scalable and flexible data center network[J]. *Communications of the ACM*, 2009, 54(3): 95-104

[4] Guo C, Lu G, Li D, et al. BCube: a high performance, server-centric network architecture for modular data centers[J]. *ACM SIGCOMM Computer Communication Review*, 2009, 39(4): 63-74

[5] 曹绍华, 卢清华, 张红霞, 等. 基于 SDN 的服务器集群动态流量调度方法[J]. 中国电子科学研究院学报, 2016, 11(6): 625-628

[6] Qazi Z A, Tu C C, Chiang L, et al. Simple-fying middlebox policy enforcement using SDN[J]. *ACM SIGCOMM Computer Communication Review*, 2013, 43(4): 27-28

[7] Al-Fares M, Radhakrishnan S, Raghavan B, et al. Hedera: dynamic flow scheduling for data center networks[C]//Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, Berkeley, USA, 2010: 19-19

[8] Benson T, Akella A, Maltz D A. Network traffic characteristics of data centers in the wild[C]// Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, New York, USA, 2010: 267-280

[9] Wang Y, You S. An efficient route management framework for load balance and overhead reduction in SDN-based data center networks[J]. *IEEE Transactions on Network and Service Management*, 2018, 15(4): 1422-1434

[10] Long H, Shen Y, Guo M, et al. LABERI-O: dynamic load-balanced routing in openflow-enabled networks[C]//IEEE International Conference on Advanced Information Networking and Applications, Barcelona, Spain, 2013: 290-297

[11] Sehery W, Clancy T C. Load balancing in data center networks with folded clos architectures[C]// Proceedings of the 2015 1st IEEE Conference on Network Softwarization, London, UK, 2015: 1-6

[12] Guo Z H, Xu Y, Liu R Y, et al. Balancing flow table occupancy and link utilization in software-defined networks[J]. *Future Generation Computer Systems*, 2018, 89: 213-223

[13] Jo E, Pan D, Liu J, et al. A simulation and emulation study of SDN-based multipath routing for fat-tree data center networks[C]//Proceedings of the Winter Simulation Conference, Savanah, USA, 2014: 3072-3083

[14] Li Y, Pan D. OpenFlow based load balancing for fat-tree networks with multipath support[C]//Proceedings of the 12th IEEE International Conference on Communications, Washington, USA, 2013: 1-5

[15] 李松州, 束永安. 基于流调度选择的 DCN 动态负载均衡算法[J]. 计算机应用研究, 2019, 36(1):205-208

[16] Mininet Team. Mininet: an instant virtual network on

your laptop (or other PC)[EB/OL]. http://mininet.org/: Mininet Team, 2019

[17] Nippon Telegraph and Telephone Corporation. Welcome to RYU the network operating system(NOS)[EB/OL]. http://ryu.readthedocs.io/en/latest/: Nippon Telegraph and Telephone Corporation, 2019

[18] Dugan J, Elliott S, Poskanzer J, et al. iPerf: the ultimate speed test tool for TCP, UDP and SCTP[EB/OL]. https://iperf.fr/: ikoula, 2019

[19] Linux Foundation. Open vSwitch[EB/OL]. http://openvswitch.org/: Linux Foundation, 2019

An efficient joint load balancing algorithm for flow tables and links in data center

Fu Qiongxiao, Sun Enchang, Wang Qianwen, Li Meng, Zhang Yanhua

(Faculty of Information Technology, Beijing University of Technology, Beijing 100124)

Abstract

In data center networks based on software defined network (SDN), mice-flows have low bandwidth, short duration and high proportion which tends to cause flow table load imbalance. On the contrary, elephant-flows tend to cause link load imbalance. According to the characteristics mentioned above, an efficient joint load balancing algorithm for flow tables and links (JLBFTL) is proposed in data center. In the algorithm, all new flows are regarded as mice-flows, the evaluation index of flow tables is introduced to evaluate and select the path for routing, realizing flow table load balance. Periodically monitoring the flows in the network, and for newly detected elephant-flows, the evaluation index of links is introduced to evaluate and select the path , realizing link load balance. In addition, a large amount of sudden traffic may lead to the overload of some links, resulting in link load imbalance. At this time, this paper proposes to realize link load balance quickly by selecting appropriate elephant-flow and using the backup path and group table for efficient triage. Simulation results show that, compared with SRL + FlowFit and L2RM, the proposed JLBFTL algorithm can reduce packet loss rate, while increase bandwidth utilization and throughput to different degrees, finally improve network performance.

Key words: software defined network (SDN), data center network, load balance for flow table, load balance for link