DOI:10.11721/cqnuj20170439

软件定义网络中集群虚拟网络的构建*

李 波, 郭松涛
(西南大学 电子信息工程学院, 重庆 400715)

摘要:【目的】随着云计算平台越来越庞大并且呈现出动态化特点,对如何满足多个用户的计算请求而进行有效资源分配,同时保障整个网络的性能进行探索。【方法】首先,提出一种集群虚拟网络(CVN)策略和动态堆排序算法。其次,将联合多个 pod 构建跨 pod 集群虚拟网络(CP-CVN),同时保障网络的性能。进而将此网络优化问题公式化为线性规划问题(LP),并提出一种近似对偶算法解决此线性规划问题。【结果】理论分析表明所提出的对偶算法是可行的,实验结果验证对偶算法解决了软件定义网络中的资源分配问题,同时保障网络性能。【结论】理论和实验分析表明所提出的对偶算法是可行的。

关键词:软件定义网络; 虚拟网络; 网络性能
中图分类号:TP31 **文献标志码**:A **文章编号**:1672-6693(2017)05-0087-07

随着大型数据中心云平台的出现,如谷歌的 B4[1] 以及亚马逊的 Amazon EC2 等,有效地"按需分配"资源并保障网络的整体性能成为巨大的挑战。云计算的目标是,租户可以根据"随收随付"模式租赁计算资源,云提供商可以管理计算资源以保证服务的可扩展性和有效性。然而,由于租户的大量服务和应用程序通常共享同一资源池,使得数据中心环境变得非常复杂。网络虚拟化旨在将物理网络资源虚拟化为逻辑网络,将这些虚拟资源提供给租户以构建自己的计算网络。此外,软件定义网络(Software defined networking,SDN)作为数据中心网络的管理者,逻辑上控制整个云计算资源,并向云提供商提供自适应策略来管理整个网络。

最近,虚拟化及 SDN 的这些优势在学术界引起了极大关注[2-6]。在文献[2]中,Ballani 提出了一种 Oktopus 系统,该系统实现虚拟网络抽象以达到租户需求的性能保证与云提供商利益之间的平衡。与 Oktopus 相反,Cloudmirror[3] 在大多数情况下采用 TAG 模型均衡带宽要求,并找到比 Oktopus 更有效的 VM 放置方法。Silva 等人[4] 提出了一种基于拓扑感知虚拟机放置算法,将每组 VM 集中放置在数据中心的小区域以实现能源效率的最大化。张顺利等人[5] 从服务管理的角度对虚拟网络资源进行分配,提出一种基于拍卖的虚拟网络资源分配机制。在文献[7]中,Rocha 等人提出一种流量感知的 VM 放置方法并将混合整数线性规划(MILP)问题公式化。本研究所使用的模型是,将 VM 聚合到 ToR 交换机所连接的服务器上的模型。文献[8]提出了一种动态的网络虚拟资源分配算法,根据节点的负载容忍度以及链路的带宽、时延进行 VM 安置。

此外,关于资源分配在减少能量消耗方面,Kliazovich 等人[9] 通过结合能量感知和网络感知因素提出一种 VM 放置方法以平衡整个数据中心的能耗。基于此方法,Kliazovich 等人[10] 通过将工作负载集中在紧密连接的服务器上计算以提高能量效率。在文献[11]中,Beloglazov 等人将 VM 放置问题建模为装箱问题,并提出了基于减少能量消耗的最佳拟合算法。Gulati 等人[12] 提出了分布式资源调度器(DRS)来管理物理资源,并将一组虚拟机布置在集群的主机上。文献[13]将待调度的资源划分为多个资源集群,建立能耗优化的资源分配模式。在文献[14-15]中,提出了两个流行的 VM 放置问题,即约束满足问题(CSP)和约束多背包问题(CMKP),以最大化计算资源的效用。虽然这些算法试图将 VM 集中在更少的服务器上,但实际上并没有考虑到服务器的通信能力。

然而,Oktopus[2] 和 Cloudmirror[3] 只考虑连接在单个虚拟交换机上的 VM 分配给租户,而没有考虑到这些 VM 在实际物理基础设施之间的距离。此外,Silva 等人[4] 旨在将 VM 组放置在尽可能小的物理设施区域中,然

* **收稿日期**:2017-03-13 **修回日期**:2017-04-20 **网络出版时间**:2017-05-16 11:25
资助项目:国家自然科学基金(No.61373179)
第一作者简介:李波,男,研究方向为软件定义网络、云计算,E-mail:413748149@163.com;**通信作者**:郭松涛,教授,E-mail:stguo@swu.edu.cn
网络出版地址:http://kns.cnki.net/kcms/detail/50.1165.N.20170516.1125.026.html

而,VM之间的通信路径不能保证是合理的。MILPFlow[6]最小化放置VM于连接在ToR的服务器上的总分配成本,但忽略了只有通过核心交换机作为桥梁才能相互通信的VM资源。文献[11,13-15]试图将VM集中在更少的服务器上,但实际上并没有考虑到服务器的通信能力。

基于以上研究,本文提出了一种新的集群虚拟网络(Clustered virtual network,CVN)抽象策略,目的是将CVN放置在连接到边缘交换机的服务器上,并且保证租户应用的服务质量(QoS)以及整个网络性能。因此,利用单个pod中的资源构建CVN以完成计算任务,这将极大地减少对网络资源的需求。此外,为了最大限度地利用资源,本研究进一步提出构建跨pod的集群虚拟网络(Crossing pod CVN,CP-CVN)的策略,以确保网络吞吐量最大化。

1 网络模型及问题阐述

1.1 网络模型

定义$v(M,N,R)$表示一个两级交换机fat-tree数据中心结构,其中M表示核心交换机的数量,它为下层的pods之间提供连接;N表示DCN中的服务器的数量;R表示边缘交换机的数量。边缘层中的每个n端口交换机通过$n/2$个端口连接到$n/2$个服务器,剩余的$n/2$个端口连接到核心交换机,每个边缘交换机及直接连接的服务器所形成网络的基本单元,称为一个pod。因此,在网络模型中共有R个pods,表示为$P=\{P_1,P_2,\cdots,P_r,\cdots,P_R\}$。

如图1的右侧部分所示,可以将SDN架构划分为3层,即控制器、协调器和云操作系统。控制器在SDN中扮演管理者角色,它收集由网络中的交换机上传的流量信息,同时使用强大的OpenFlow协议与交换机通信。云操作系统,例如OpenStack,管理所有计算资源(服务器或VM)。所有的VM周期性地向云操作系统报告最新的状态,即空闲或在用。在控制器和云操作系统的帮助下,协调器组合收集的信息并对VM的通信做出适当的决定。

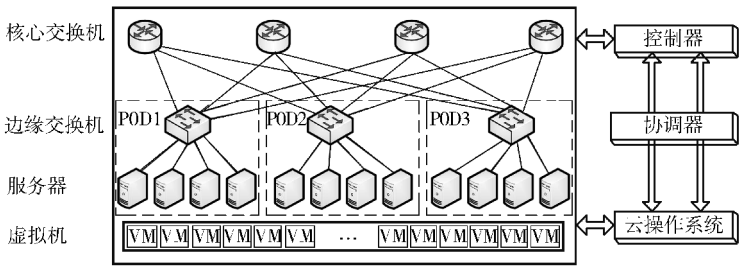

图1 网络模型

Fig. 1 Network model

1.2 集群虚拟网络抽象

一般来说,云供应商提供的物理资源,租户不能直接操作这些资源。因此,云供应商倾向于虚拟化这些物理资源,同时对这些虚拟资源进行集中管理,只需向租户提供租赁资源的抽象接口。租户只需要关心的是租赁资源的类型和数量,即CPU的时隙、内存大小和分配的链路带宽等。

每个VM之间共享公共物理硬件资源。为了简化描述,假设硬件资源分配给每个VM的资源是平均的。在网络模型中,如果VM没有租赁给租户,则称之为空闲VM,用$F=\{F_1,F_2,\cdots,F_r,\cdots\}$表示空闲VM的集合,其中$F_r$表示$P_r$中的空闲VM集合,$1\leqslant r\leqslant R$。此外,如果VM已经被租用,并且没有释放,则称为在用VM,用$U=\{U_1,U_2,\cdots,U_r,\cdots,\}$表示在用VM的集合,其中$U_r$是在$P_r$中使用的VM的集合,$1\leqslant r\leqslant R$。

集群虚拟网络可以为租户提供可管理的计算资源(如集群VM和足够的链路带宽)完成计算任务,这些计算资源放在一个pod中。然而,为了实现这一目标,必须解决几个具有挑战性的问题。第一个问题是VM分配策略,即找到具有最适当数量的虚拟机(VM)的pod并且在一个pod中快速地在相应的VM上分配计算任务。第二个挑战是,如果在任何一个pod中没有足够的VM,云提供商应该能够构造CP-CVN,以便最大限度地利用资源为租户提供服务,同时保证网络性能。

2 CVN的构建

2.1 虚拟机分组

首先,根据它们所在的pod将所有VM分成R组,并且假设在第r个pod中最初有空闲的P_r个VM。然后,每个VM定期向云操作系统上传最新状态,即空闲或在用。云操作系统根据上传的信息检查或修改云数据库中的VM的状态。以这种方式,根据它们的状态(空闲或在用),第r个pod中的所有VM将被分成两组,分别

用 F_r 和 U_r 表示。当协调器接收到计算任务请求时，它将在整个资源池搜索具有最合适数量 VM 的 pod，并在该 pod 中分配 VM 以执行任务。

2.2 寻找最合适的 pod

为了找到具有最适当数量的空闲虚拟机的 pod，应该对集合 $F=\{F_1,F_2,\cdots,F_R\}$ 的所有元素进行排序。可以通过改进的堆算法将集合 F 的所有元素进行排序构建二进制堆。为了方便，定义数组 $A[1],A[2],\cdots,A[R]$ 表示一个堆，它的元素 $F_1,F_2,\cdots,F_R$ 具有部分有序树的属性，即

$$\begin{cases}A[r]\geqslant A[2r],\\A[r]\geqslant A[2r+1]。\end{cases}\tag{1}$$

这意味着数组形成一个平衡二叉树，其中子节点的值小于或等于父节点的值。

为了避免违反部分有序树的属性，提出了一个动态 heapsort 算法，使用 pushdown 算法[16]将数组 A 构造成具有完全有序树属性的堆。具体地，假设数组 $A[a],\cdots,A[b]$ 的所有元素具有除了 $A[a]$ 的子节点之外的部分有序树的属性。通过一系列交换，pushdown 算法将元素 $A[a]$ 放置至树中的适当位置。算法 1 给出了寻找具有最适当数量的空闲 VM 的 pod 的算法。

在动态堆排序算法（表 1）的帮助下，云操作系统获得递增或递减的元素 $F_1,F_2,\cdots,F_R$ 排列。然后，协调器将相应数量的 VM 与固定顺序集合 F 的元素 F_r 相匹配，其中 $1\leqslant r\leqslant R$。因此，协调器一旦收到任务请求，便可以更快地找到具有最适当数目空闲 VM 的 pod 为租户构建 CVN。然而，当有新的计算任务到达时，可能在一个 pod 中并不总是存在足够的空闲 VM，但可以联合多个 pod 的资源进行提供服务。

表 1　动态堆排序算法

Tab. 1　Dynamical heapsort algorithm

算法 1　动态堆排序算法伪代码
输入： 每一个 pod 内的空闲虚拟机数目：$F_1,F_2,\cdots,F_r$，$1\leqslant r\leqslant R$； pod 的总数量 R。
输出： 降序排列的数组 $A[1],A[2],\cdots,A[R]$。
1：初始化整数变量 r； 2：for $1\leqslant r\leqslant R/2$ do 3：调用 *pushdown*(r,R)； 4：end for 5：for $2\leqslant r\leqslant R$ do 6：交换$(A[1],A[r])$的位置； 7：调用 *pushdown*$(1,r-1)$； 8：end for 9：更新数组 A 中每一个 pod 的数值。

3 网络性能优化

3.1 构建 CP-CVN

首先，假设每一个 pod 内都有一个非阻塞边缘交换机连接这个 pod 中的所有服务器。因此，不需考虑在单个 pod 中构造 CVN 的链路容量[17-18]。然而，对于不同 pod 中的 VM 的通信，需通过核心交换机进行数据交换。为了减少核心交换机的流量负载，最好将 CP-CVN 的所有成员限制在较少的 pod 中。为此，提出一个 max-min 策略，即协调器首先选择具有最大数量的空闲 VM 的 pod，然后从剩余 pod 中找到具有最少空闲 VM 的 pod，使得最大和最小的 pod 内空闲 VM 数目达到计算任务所需的虚拟机数量。因此，为了构建具有最大资源效用的 CP-CVN，当不同 CP-CVN 中的 VM 相互通信时，有效地利用核心交换机之间的链路带宽成为焦点。接下来，将提出一个近似算法以最大化网络吞吐量，实现最佳资源效用。

3.2 最大化网络吞吐量

CP-CVN 不仅需要使用 pod 中内部资源，而且还需要外部资源，如核心交换机和链路。因此，为了获得最大的网络流，提出了一种近似算法，以在固定的链路容量的条件下最大化整个网络吞吐量。

3.2.1 *近似算法*　首先，对网络吞吐量最大化问题进行公式化。将软件定义网络（SDN）模型化为图 $G(V,E)$，其中 V 表示节点集合，E 表示节点之间的链路集合。假设有一对流从源到目的，$(s_1,d_1),\cdots,(s_I,d_I)$。另外，使 $c(e)$ 和 $l(e)$ 分别表示链路 $e\in E$ 的容量和通信成本，即分配的带宽。假设 $l_i(e)$ 是第 i 对流通过链路 e 的通信成本，其中 $e\in E$。m 表示网络 G 中的链路的数量，P_i 表示从源 s_i 到目的 d_i 之间的路径集合，$i=1,2,\cdots,I$。P 指 P_i 的集合，表示为 $P=\{P_i\mid i=1,2,\cdots,I\}$。此外，令 P_e 为路径 P 中所有使用链路 e 的路径集合，所有 $e\in E$，$x(p)$ 表示路径 $p\in P$ 中的流。因此，可以将网络吞吐量最大化问题表示为：

$$\max \sum_{p\in P} x(p) \tag{2}$$

$$\text{s.t.} \sum_{p\in P_e} x(p) \leqslant c(e), \forall e \in E, \tag{3}$$

$$x(p) \geqslant 0, \forall p \in P, \tag{4}$$

$$l(e) = \sum_{i=1}^{I} l_i(e), \forall e \in E, \tag{5}$$

$$l_i(p) = \sum_{e\in p} l_i(e), \forall p \in P, i = 1,2,\cdots,I, \tag{6}$$

$$x(p) = \sum_{i=1}^{I} l_i(p), \forall p \in P。 \tag{7}$$

其中,容量约束(3)保证链路 e 上的所有流的总和不能超过链路 e 的容量 $c(e)$,对于所有 $e\in E$。约束(4)确保路径 p 上的流 $x(p)$都为非负值,$p\in P$。(5)式表示链路 e 的通信成本等于该链路上所有流的通信成本之和,$e\in E$。在(6)式中,令 $l_i(p)=\sum_{e\in P} l_i(e)$ 为对于第 i 对流通过路径 p 的通信成本,$p\in P$,$i=1,2,\cdots,I$。此外,在(7)式中,$x(p)$表示所有经过路径 p 的数据流的通信成本,$p\in P$。

显然,问题(2)是一个线性规划问题,它的对偶形式可写为:

$$\min D(l) \overset{\text{def}}{=} \sum_{e\in E} c(e)\cdot l(e) \tag{8}$$

$$\text{s.t.} \sum_{e\in p} l(e) \geqslant 1, \forall p \in P;$$

$$l(e) \geqslant 0, \forall e \in E;$$

$$l(e) = \sum_{i=1}^{I} l_i(e), \forall e \in E。$$

其中,通信代价 $l(e)$可以看作该线性规划问题的对偶变量。

不难观察到求解线性规划问题(2)和(8)式具有指数时间复杂度。因此,在算法2中给出了近似算法(表2)解决该线性规划问题,根据 G 中的链路数 m 和精度因子 δ,将链路 e 的的成本 $l(e)$设置为 $\dot{o}$,$e\in E$,并且对于所有 $p\in P$,将 $x(p)$设置为0。路径 p 上的流量由该路径的最小容量 c 限制,因此 $x(p)$由 $x(p)=x(p)+c$ 更新。该算法迭代直到找到最大数目的流和合适的对偶变量时终止。

3.2.2 *算法分析* 对于给定的链路成本 l,为了呈现方便,定义 $\alpha(l)\overset{\text{def}}{=}\min_{p\in P} l(p)$,$\beta\overset{\text{def}}{=}\min_{l} D(l)/\alpha(l)$。令 l_{i-1}表示第 i 次迭代开始时的链路代价函数,f_{i-1}表示从第1次到第 $i-1$ 次迭代全部的流量。进一步,为了表示的方便性,定义 $\alpha(i)$和 $D(i)$分别表示 $\alpha(l_i)$和 $D(l_i)$。算法在 t 次迭代之后终止,其中 t 是使得 $\alpha(t)\geqslant 1$ 的最小值。

引理1 迭代 $1-\dfrac{\log\delta}{\log(1+\dot{o})}$次后得到的最终数据流是可行的数据流。

证明 假设每一次迭代链路上的全部数据流增加了链路容量的分数 $a_i(0<a_i<1)$,该链路的成本乘以 $1+a_i\dot{o}$。因为$(1+\dot{o})^a\geqslant 1+a\dot{o}$,$\forall a\in[0,1]$,因此可以得到 $\prod_i(1+a_i\dot{o})\geqslant(1+\dot{o})^{\sum_i a_i}$,其中对于所有的 i 有 $a_i\in[0,1]$。因此,可以观察到链路的成本增加至少$(1+\dot{o})$因子,并且链路上的总流量根据相应的容量增加。因为 $l(e)$的初始值设置为 δ,迭代终止时 $l(e)<1+\dot{o}$,因此链路 e 上的数据流不会超过$c(e)\cdot\log_{1+\dot{o}}\dfrac{1+\dot{o}}{\delta}$。 证毕

表2 近似算法

Tab.2 Approximation algorithm

算法2 近似算法伪代码
输入:
软件定义网络 $G(V,E)$;
链路容量 $c(e)$,$e\in E$;
数据流对(s_i,d_i),$1\leqslant i\leqslant I$;
因子 $\dot{o}$。
输出:
路径 p 的所有数据流的和 $x(p)$,$p\in P$;
对偶解 $l(e)$,$e\in E$。
1:初始化 $l(e)=\delta$,$\forall e\in E$;
2:初始化 $x(p)=0$,$\forall p\in P$;
3:for $1\leqslant i\leqslant(1-\log\delta/\log(1+\dot{o}))$ do;
4:for $1\leqslant j\leqslant I$ do;
5:p:在 P_j 中使用 l 的最短的合理路径;
6:while $l(p)<\min\{1,\delta(1+\dot{o})^i\}$ do;
7:$c=\min_{e\in P} c(e)$;
8:$x(p)=x(p)+c$;
9:$l(e)=l(e)\left(1+\dfrac{c\dot{o}}{c(e)}\right)$,$\forall e\in p$;
10:end while;
11:end for;
12:end for;
13:return $x(p)$,$\forall p\in P$;
14:return $l(e)$,$\forall e\in E$。

4 实验仿真

4.1 仿真环境搭建

Mininet 是一个被广泛肯定和采用的网络仿真器，它使用轻量级虚拟化在单个系统（如计算机或虚拟机）上部署大型网络。如图 2 所示，对$v(16,512,16)$fat-tree拓扑结构的数据中心网络进行了模拟。其中，采用 $n=64$ 端口交换机和共有 $N=512$ 个服务器，每个服务器包含 20 个 VM（总共 10 240 个 VM）。服务器和边缘交换机之间的链路带宽为 1 Gbps，而核心交换机和边缘交换机之间的链路带宽为 10 Gbps。为了评估所提出的算法的有效性（主要对网络吞吐量指标进行评估），与前面提到的 Oktopus，CloudMirror 和 MILPFlow 算法进行比较。接下来，将详细地描述更多仿真条件设置，如通信模式、资源分配和流生成等。

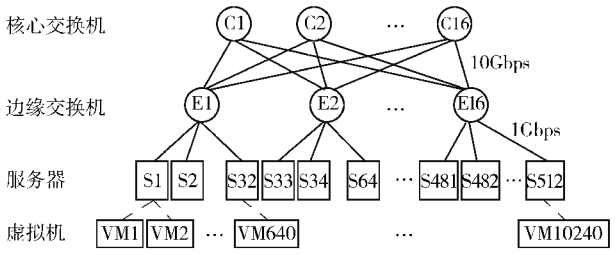

图 2　fat-tree 数据中心结构

Fig. 2　Fat-tree data center

4.1.1 通信模式　在多重通信模式下，对于租户的 CVN，其成员内通信模式会存在这样的情况，多个 VMi 向任何其他 VM $x+i$ 发送数据流，并且租户的源和目的 VM 可以分配在不同的 pod 中。为了模拟更真实的云环境，设计了 40 个租户同时共享同一资源池中的资源，并设置每个租户租用的虚拟机数量是 180～200 之间的随机偶数。在这种情况下，根据多重通信模式，每个租户的 VM 将被平均地划分为两种，即源和目的 VM。

4.1.2 资源分配　基于多重通信模式，在 3 种资源分配模式下进行模拟仿真实验：1）单个 pod。资源 VM i 和目的 VM $x+i$ 被分配在同一个 pod 中。将一个 pod 的 VM 分配给不超过 3 个租户；2）不同的 pods。所有源 VM 被分配在与目的 VM 位置不同的 pod 中；3）组合情况。情况 1）和情况 2）的组合，将大约 1/3 租户的租用虚拟机设置为不同的 pod 模式，其他租户的租用虚拟机保持单个 pod 模式。

4.1.3 数据流生成　另一方面，为了使模拟仿真结果更具有说服力，所有流都是根据实际数据中心网络中自然生成，且流的大小随机。图 3 和图 4 分别给出了流的大小分布的概率密度函数（PDF）和累积分布函数（CDF）。从数据流大小的 PDF 和 CDF 可以看出，几乎所有流均小于 100 MB，大约 90％的字节在 100 MB 和 1 GB 之间的数据流中。考虑到公平性，所产生的数据流都是按照统一策略并且数据流的数目也是相同的。最终结果取自于 100 次实验所得数据的平均值。

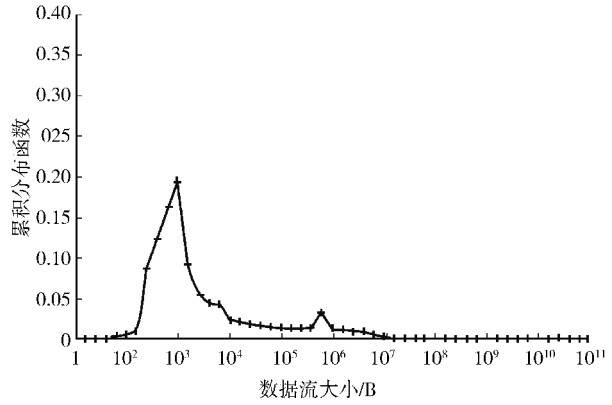

图 3　数据流大小的概率函数分布

Fig. 3　PDF of the distribution of flow size

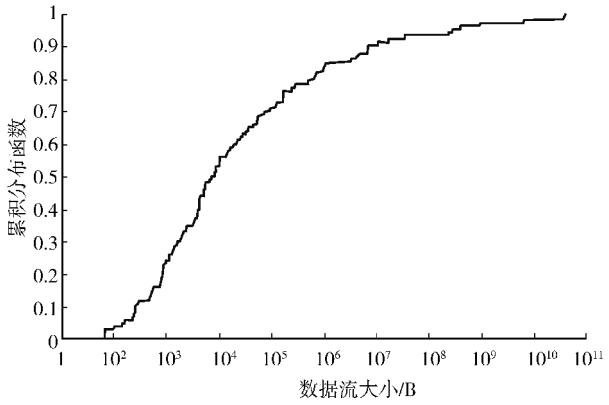

图 4　数据流大小的累积函数分布

Fig. 4　CDF of the distribution of flow size

4.2 实验分析

根据多重通信模式，首先模拟单个 pod 内的资源分配。考虑到实际情况，为了避免流同步生成，规定生成的每个流之间时间间隔 100 ms。因此，整个生成流的过程将在 10 s 内完成。在相同的条件下，本研究提出的算法与其他 3 种资源分配算法分别布置在 fat-tree 网络中，为了方便说明，现将提出的算法表示为 CVN 算法。图 5 显示了 CVN 算法和 CloudMirror，Oktopus 和 MILPFlow 算法的网络吞吐量性能比较，其中源和目的 VM 分布在单个 pod 中。显然，对于所有算法，网络吞吐量在 15 s 内不断增加，然后趋于平稳。仿真结果表明，CVN 算法的网络吞吐量性能优于 CloudMirror，Oktopus 算法 6％，优于 MILPFlow 算法 10％，这主要是因为 CVN 算法是

基于所有链路的状态信息做出自适应链路分配决定,同时最小化带宽使用。

此外,为了进一步评估CVN算法的性能,采用VM分配在不同pod内的资源分配策略。图6中的结果表明,当源和目的VM分布在不同的pod中时,CVN算法的性能分别比CloudMirror,Oktopus和MILPFlow的性能提高了约22%,25%和28%。这是因为CVN算法很好地利用不同pod之间的网络资源,为VM之间的通信建立多跳路径并最大化路径上的数据流。此外,本研究还将前两种情况结合起来,同时构建CVN和CP-CVN以满足租户的需求。实验的结果如图7所示,CVN算法的性能分别比CloudMirror,Oktopus和MILPFlow算法的性能优15%,18%和32%。综合以上的结果,可以得出CloudMirror,Oktopus和MILPFlow算法只考虑从单个pod中分配VM资源,并没有同时考虑不同pod的资源的联合。

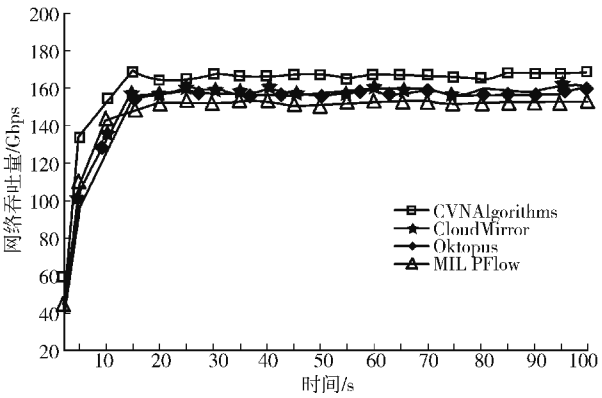

图5 单个pod情况下的网络吞吐量

Fig. 5 Aggregation throughput under the pattern of single pod

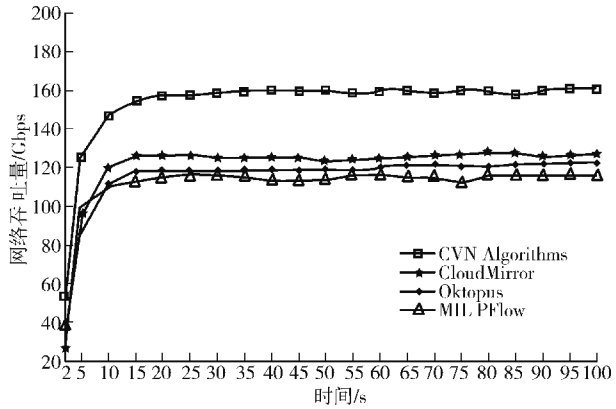

图6 不同pod情况下的网络吞吐量

Fig. 6 Aggregation throughput under the pattern of different pods

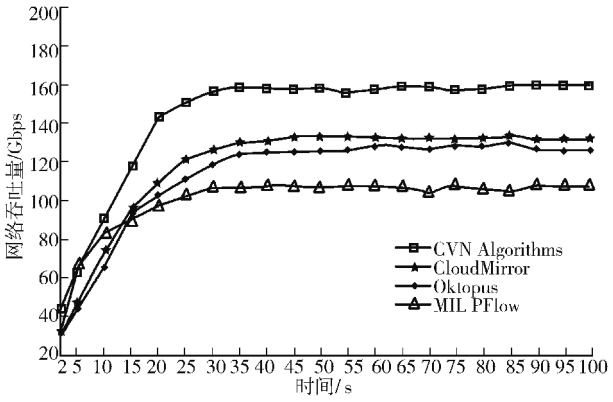

图7 混合模式下的网络吞吐量

Fig. 7 Aggregation throughput under the pattern of combined case

5 结论及展望

本研究主要解决了基于SDN的fat-tree数据中心网络中的资源分配问题。首先,提出了集群虚拟网络(CVN)抽象策略。其中,包含两个步骤:第一,通过动态的堆排序算法寻找到具有最合适数目VM的pod进行资源分配;第二,构建跨pod的CP-CVN以最大化资源利用率,同时保障网络的整体性能。进一步,将次网络优化问题公式化为对偶问题,并提出一种近似的算法解决。实验结果表明,在不同pod情况下,本研究所提出的CVN算法与CloudMirror,Oktopus和MILPFlow算法相比网络吞吐量分别提高了22%,25%和28%;在混合模式下,网络吞吐量分别提高了15%,18%和32%。

本研究主要致力于解决单SDN控制器的数据中心网络的资源分配问题。随着云计算的快速发展,越来越多的数据中心分布在世界的各个区域,这就需要多个控制器同时管理云计算资源。因此,在未来的云计算环境中,数据中心的多控制器共同协作以完成资源分配问题将变得更具挑战性,下一步,将继续研究多控制器协同工作下的资源分配策略。

参考文献:

[1] JAIN S,KUMAR A,MANDAL S,et al.B4:experience with a globally-deployed software defined wan[C]//ACM SIGCOMM 2013.Hong Kong:ACM,2013.

[2] BALLANI H,COSTA P,KARAGIANNIS T,et al.Towards predictable datacenter networks[C]//ACM SIGCOMM Computer Communication Review.America:ACM,2011,41(4):242-253.

[3] LEE J,LEE M,POPA L,et al.CloudMirror:application-aware bandwidth reservations in the cloud[C]//InHotCloud.Princeton:Citeseer,2013.

[4] SILVA R A C D,FONSECA N L S D.Topology-aware virtual machine placement in data centers[J].Journal of Grid Computing,2016,14(1):75-90.

[5] 张顺利，邱雪松，陈东东，等.网络虚拟化环境中虚拟网资源分配机制[J].北京邮电大学学报,2011,34(6):24-27.
ZHANG S L,QIU X S,CHEN D D,et al.Virtual network resource allocation mechanism in network virtualization environment[J].Journal of Beijing University of Posts and Telecommunications,2011,34(6):24-27.

[6] 曹傧,郎文强,李云.基于最大收益的无线虚拟网络重映射算法[J].重庆邮电大学学报(自然科学版),2016,28(5):620-627.
CAO B,LANG W Q,LI Y.Virtual network reconfiguration in wireless network virtualization based on maximum revenue[J].Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition),2016,28(5):620-627.

[7] ROCHA L,VERDI F.A network-aware optimization for VM placement[C]//2015 IEEE 29th international conference on IEEE.California:IEEE,2015.

[8] 罗娟,徐岳阳,李仁发.网络虚拟化中动态资源分配算法研究[C]//通信学报学术论坛暨2011云计算学术会议.北京:通信学报,2011.
LUO J,XU Y Y,LI R F.Research on dynamic resource allocation algorithm in network virtualization[C]//Journal of communication and academic forum and 2011 cloud computing conference.Beijing:Journal on Communications,2011.

[9] KLIAZOVICH D,BOUVRY P,KHAN S U.DENS:data center energy-efficient network-aware scheduling[J].Cluster Computing,2013,16(1):65-75.

[10] KLIAZOVICH D,ARZO S T,GRANELLI F,et al.e-STAB:energy-efficient scheduling for cloud computing applications with traffic load balancing[C]//IEEE international conference on green computing and communications.Beijing:IEEE,2013.

[11] BELOGLAZOV A,ABAWAJY J,BUYYA R.Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing[J].Future Generation Computer Systems,2012,28(5):755-768.

[12] GULATI A,HOLLER A,JI M,et al.Vmware distributed resource management:design,implementation,and lessons learned[J].VMware Technical Journal,2012,1(1):45-64.

[13] 林伟伟,朱朝悦.面向大规模云资源调度的可扩展分布式调度方法[J].计算机工程与科学,2015,37(11):1997-2005.
LIN W W,ZHU C Y.Scalable distributed scheduling method for large scale cloud resource scheduling[J].Computer Engineering and Science,2015,37(11):1997-2005.

[14] VAN H N,TRAN F D,MENAUD J M.Autonomic virtual resource management for service hosting platforms[C]//In proc ICSE workshop on software engineering challenges of cloud computing.Washington,DC:IEEE,2009.

[15] TANG C,STEINDER M,SPREITZER M,et al.A scalable application placement controller for enterprise data centers[C]//International conference on World Wide Web.New York:ACM,2007.

[16] AHO A V,HOPCROFT J E,ULLMAN J.Data structures and algorithms[J].Nature,1983,23(3):27-41.

[17] GUO Z,DUAN J,YANG Y.Oversubscription bounded multicast scheduling in fat-tree data center networks[J].IEEE 27th International Parallel and Distributed Processing Symposium,2013:589-600.

[18] SHAHROKHI F,MATULA D W.The maximum concurrent flow problem[J].Journal of the Acm,1990,37(2):318-334.

Construction of Clustered Virtual Network in Software Defined Networks

LI Bo, GUO Songtao

(College of Electronic and Information Engineering, Southwest University, Chongqing 400715, China)

Abstract: [**Purposes**]As cloud computing becomes more and more massive and dynamic, it is a great challenge to effectively allocate resource for different tenants' computing requests while considering the performance of the whole network. [**Methods**]To address this problem, they propose a clustered virtual network (CVN) abstraction strategy and a dynamical heapsort algorithm. Subsequently, they construct a crossing pod CVN (CP-CVN) and guarantee network performance. Furthermore, they formulate the network optimization problem as a Linear Programming (LP) problem. To achieve computation feasibility in massive data center networks, they propose an approximation primal-dual algorithm for solving this LP problem. [**Findings**]Theoretical analysis shows that the proposed primal-dual algorithm is feasible and the experiment results verify that the algorithm is suitable for solving massive computation problem in SDNs. [**Conclusions**]Theoretical and experimental analysis shows that the proposed primal-dual algorithm is feasible and suitable.

Keywords: software defined networking; virtual network; network performance

(责任编辑 游中胜)