激光与光电子学进展

弹性光数据中心中软管虚拟机的放置算法研究

马中俊，刘逢清*，陈宇星

南京邮电大学电子与光学工程学院、柔性电子(未来技术)学院，江苏 南京 210023

摘要 为了吸纳业务的不确定性并将业务网络灵活高效地映射到数据中心的物理网络中，研究了弹性光数据中心网络中业务模型为软管模型的动态虚拟数据中心映射问题。首先建立了软管虚拟机在弹性光数据中心网络中的映射模型，然后提出了基于虚拟拓扑图的虚拟机放置算法(VT-VMPA)。VT-VMPA首先将软管模型转化为管道模型，再由业务量最大化原则寻找核心虚拟机。然后按照业务量降序，将满足资源约束条件且与核心虚拟机相连的虚拟机合并成簇，以减少簇映射到服务器后的通信带宽需求。最后将构成簇集按跳距自适应和最短路径原则映射到弹性光数据中心网络的服务器和光路上。仿真结果显示，该算法与其他算法相比，平均带宽消耗减少了21%、阻塞率减少了27%、时间平均收益提高了117%。这表明该算法能够降低网络带宽资源的消耗、提高网络映射率。

关键词 光通信；弹性光网络；软管模型；虚拟网络映射；虚拟机放置

中图分类号 TN929.11 **文献标志码** A **DOI**：10.3788/LOP222310

Research on Placement Algorithm of Flexible Virtual Machine in Elastic Optical Data Center

Ma Zhongjun, Liu Fengqing*, Chen Yuxing

School of Electronic and Optical Engineering, School of Flexible Electronics (Future Technology), Nanjing University of Posts and Telecommunications, Nanjing 210023, *Jiangsu, China*

Abstract To absorb the traffic uncertainty and map the service network to the physical network of the data center flexibly and efficiently, this paper studies the dynamic virtual data center mapping problem in an elastic optical data center network, where the service model is a hose model. First, the mapping model of a flexible virtual machine is established in an elastic optical data center network, and then the virtual machine placement algorithm based on virtual topology (VT-VMPA) is proposed. The VT-VMPA first converts the hose model into the pipe model before looking for the core virtual machine using the maximization of traffic volume as a guideline. Then, in descending order of service volume, the virtual machines that adhere to the resource constraints and are linked to the core virtual machines are combined into clusters. The communication bandwidth requirement is reduced after the cluster is mapped to the server. Finally, the cluster sets are mapped to the servers and virtual links between clusters are mapped to optical paths of the elastic optical data center network according to hop distance adaptive and shortest path principles. In comparison to previous methods, the proposed algorithm has decreased blocking rate by 27%, increased average time revenue by 117%, and decreased average bandwidth usage by 21%. It demonstrates that this method may increase network mapping speed while consuming fewer network traffic.

Key words optical communication; elastic optical network; hose model; virtual network mapping; virtual machine placement

1 引 言

大数据应用和云计算技术的出现使得用户终端间的数据流通信模式越来越难以预测。文献[1]创造性地提出了一种灵活网络模型(即：软管模型)来吸收数据流的不确定性。与管道模型相比，软管模型用户带宽需求规范容易，只需要定义端点传入和传出的带宽需求，数据流更加灵活，能实现带宽的高复用增益等。但是，软管模型产生的虚拟链路带宽大小不一。因此，传统的传输波分复用网络难以满足需求，新的传输网

收稿日期：2022-08-15；**修回日期**：2022-09-11；**录用日期**：2022-09-26；**网络首发日期**：2022-10-08

基金项目：国家自然科学基金（62004105）

通信作者：*liufq@njupt.edu.cn

络——弹性光网络[2]，走进人们的视野当中。与传统的波分复用网络相比，弹性光网络解决了以固定波长大小给业务分配带宽的缺点，它可以根据业务的需求，进行频隙组合，提供超波长通道或子波长通道来进行业务传送，从而提高频谱利用率[3-5]。因此，以弹性光网络作为物理网络可以满足大容量和不同带宽颗粒的业务传输需求。为了实现客户端网络到物理网络的灵活高效映射，研究者们提出了网络虚拟化方法。它通过网络软硬件资源的抽象化，为客户网络提供统一的虚拟化资源，网络运营者可以通过虚拟化将多个客户端网络有效地嵌入到共享的物理网络中[6]。

Sahoo等[7]提出了一种优先考虑物理链路带宽资源的启发式算法。通过设定权值公式，将物理链路的跳数、剩余带宽容量等因素纳入考虑范围，从而提高执行效率。文献[8]提出了采用切比雪夫尺度化函数基于强化学习的放置算法，用来解决现有采用启发式的虚拟机放置算法如何适当选取的权值公式的问题。也有不少研究者同时考虑虚拟机(VM)间的业务需求和物理机的资源利用率，提出了一种基于簇的动态虚拟数据中心映射(VDCM)算法。这类算法采用分簇算法将虚拟机集分为不同子集的虚拟机簇，之后为每一个虚拟机簇集寻找合适的物理机，例如文献[9]～[12]。文献[13]针对虚拟机放置问题，提出了一种能考虑任务的完工时间和物理机的能耗及空闲时间的放置算法。通过将问题建模为一个三维的装箱问题，来最小化能源消耗。文献[14]提出了一种再平衡方案，以实现宿主机之间的负载均衡，用来解决VM为云用户提供服务时被频繁创建和删除而带来负载均衡问题。文献[15]通过优化放置虚拟机的宿主机位置，以兼顾宿主机的资源利用率和虚拟机通信所使用的物理链路的带宽，提高整体的资源利用率。Rugwiro等[16]提出了一种基于K-Means的虚拟机放置算法，该算法同时考虑请求的拓扑属性与业务量，计算对外业务量最大的VM，按照对外业务量降序，将虚拟机集合分为K族簇，用于解决上述问题。但是该算法的不足之处在于分簇后每个簇中虚拟机数量不能预期，有的簇虚拟机的数量过多，有的簇虚拟机的数量过少，即难以预期分簇后簇的大小对放置结果产生的影响。这样不仅会导致开启的服务器的数量增多，还会造成簇间的业务量占用网络中过多的带宽资源。

为了弥补分簇不均和上述算法的不足，同时对物理资源的合理利用，本文考虑了业务量、资源消耗、阻塞率和平均收益等参数，研究了虚拟网络中VM放置不同带来的影响，建立合适的模型。本文主要做了两部分工作：1) 建立物理弹性光数据中心网络和软管模型虚拟数据中心请求网络模型；2) 提出了灵活栅格光网络中软管模型下基于虚拟拓扑图的虚拟机放置算法(VT-VMPA)。

2 虚拟机放置问题描述

2.1 网络建模

本文的网络模型分为物理弹性光数据中心网络和软管模型虚拟数据中心请求网络。弹性光数据中心间网络建模为一无向权重图$G^{\mathrm{S}}=\left\{N^{\mathrm{S}}\left(N_{\mathrm{T}}^{\mathrm{S}},N_{\mathrm{DC}}^{\mathrm{S}}\right),L^{\mathrm{S}},R\left(N^{\mathrm{S}}\right),F_{\mathrm{S}}\left(L^{\mathrm{S}}\right)\right\}$，其中：$N^{\mathrm{S}}$是网络中节点集合，包括转换节点$N_{\mathrm{T}}^{\mathrm{S}}$和数据中心节点$N_{\mathrm{DC}}^{\mathrm{S}}$；$L^{S}$是物理节点间的链路集合；$R\left(N^{\mathrm{S}}\right)$是物理网络节点的一组可用计算资源；$F_{\mathrm{S}}\left(L^{\mathrm{S}}\right)$是物理链路中的一组频隙时隙状态。物理弹性光数据中心网络是由Fat-Tree数据中心拓扑构成，用集合$S=\left\{S_1,S_2,\cdots,S_N\right\}$表示其服务器集合。对于集合$S$中的每台服务器$S_i$，用$P_{\mathrm{CPU},i}=\left\{P_{\mathrm{cpu}}\right\}$表示CPU可用的计算容量。在数据中心拓扑中，每一对服务器S_i、S_k之间的带宽资源为$B_{i,k}$。

软管模型虚拟网络建模$G_{\mathrm{H}}^{v}=\left\{N_{\mathrm{H}}^{v},L_{\mathrm{H}}^{v},R_{\mathrm{H}}\left(N^{v}\right),B_{\mathrm{H}}\left(N^{v}\right)\right\}$[17]。在模型中，$v$表示虚拟节点，$N_{\mathrm{H}}^{v}$是用户请求中的虚拟节点集，$L_{\mathrm{H}}^{v}$是请求中两节点相连的链路集，$R_{\mathrm{H}}\left(N^{v}\right)$是软管模型用户请求节点所需的CPU计算资源集，$B_{\mathrm{H}}\left(N^{v}\right)$为用户请求中节点所需的传入或传出的软管带宽集。部署时，通过CPLEX处理软管模型用户请求，将请求中节点所需的软管带宽集以虚拟链路带宽和最小化为目标转化为通用虚拟链路带宽，形成管道模型用户请求并建模为$G^{v}=\left\{N^{v},L^{v},R\left(N^{v}\right),B\left(L^{v}\right)\right\}$，其中：$N^{v}$表示请求中虚拟节点集合；$L^{v}$表示是虚拟节点间的有向链路集合；$R\left(N^{v}\right)$表示虚拟节点所需的CPU计算资源集合；$B\left(L^{v}\right)$表示管道模型虚拟请求节点间有向链路所需的带宽资源集合。

在软管模型虚拟数据中心请求转化管道模型虚拟数据中心请求后，其虚拟数据中心(VDC)将由M台VM构成，用集合$V=\left\{v_0,v_2,\cdots,v_{M-1}\right\}$来表示。对于集合$V$中的每台虚拟机$v_j$，用$R_j=\left\{R_{\mathrm{cpu}}\right\}$表示其资源需求量。在转化后的管道模型虚拟数据中心中，每一对虚拟机v_j、v_k之间有业务量需求$f_{j,k}$，整个虚拟数据中心虚拟机对间的业务量将构成流量需求矩阵表示为

$$\boldsymbol{F}=\begin{bmatrix} f_{0,0} & \cdots & f_{0,M-1} \\ \vdots & & \vdots \\ f_{M-1,0} & \cdots & f_{M-1,M-1} \end{bmatrix},$$
$$\begin{cases} f_{j,k}=f_{k,j}, & j\neq k,0\leqslant j,k\leqslant M-1 \\ f_{j,k}=0, & j=k,0\leqslant j,k\leqslant M-1 \end{cases}。\tag{1}$$

2.2 性能评价指标

本文是以虚拟数据中心请求的平均带宽消耗、阻塞率和平均收益来评判算法的性能。

1)平均带宽消耗。在保持映射个数不变时，虚拟机之间业务量越少，消耗的平均带宽越少。

$$\frac{\min\sum_{j=0}^{M-1}\sum_{k=j+1}^{M-1} f_{j,k}\cdot A(v_j,v_k)\cdot b\left[m(v_j),m(v_k)\right]}{C}, \tag{2}$$

$$A(v_j,v_k)=\begin{cases}0 & \text{, virtual machines } v_j,v_k \text{ are deployed on the same server}\\ 1 & \text{, virtual machines } v_j,v_k \text{ are placed on different servers}\end{cases}, \tag{3}$$

式中：$f_{j,k}$表示虚拟机v_j、v_k之间的业务量需求；$A(v_j,v_k)$是指示函数，其具体含义见式(3)；$m(v_j)$与$m(v_k)$为映射函数；$b\left[m(v_j),m(v_k)\right]$为调用函数，其含义如表1所示；$C$为映射个数定值。

2)阻塞率。一段时间内阻塞的虚拟网络请求数N^t_{blocking}与虚拟网络请求总数N^t_{total}的比值

$$P^t_{\text{blocking}}=\frac{N^t_{\text{blocking}}}{N^t_{\text{total}}}。 \tag{4}$$

3)平均收益。时间平均收益定义为一段时间内平均成功映射的虚拟网络请求总收益。

$$A^t_{\text{Revenue}}=\frac{\sum_{n_{\text{vnr}}\in n^t_{\text{vnr}}} T_{n_{\text{vnr}}}\times\sum_{v\in N^v_{n_{\text{vnr}}}}\left[R(v)+B(v)\right]}{t}, \tag{5}$$

式中：n^t_{vnr}是一段时间内映射成功的虚拟网络请求集；$R(v)$是虚拟节点v所需的CPU计算量；$B(v)$是虚拟节点v所需带宽资源；$T_{n_{\text{vnr}}}$是虚拟网络请求n_{vnr}的持续时间；$N^v_{n_{\text{vnr}}}$是虚拟网络请求n_{vnr}的虚拟节点集。

表1　变量含义列表

Table 1　List of variable meanings

Variable name	Meaning
M	Total number of virtual machines in virtual center requests
Vs_i	The number of virtual machines deployed on the server S_i
$f_{x,y}$	Traffic between virtual machines x and y
$m(v_j),m(v_k)$	Map function, returns v_j、v_k, mapped physical server
$b\left[m(v_j),m(v_k)\right]$	Call a function that returns the hop distance between the physical server $m(v_j)$ and $m(v_k)$ to be mapped by the virtual machine to v_j and v_k
n^t_{vnr}	The set of virtual network requests that have been successfully mapped over a period of time
$T_{n_{\text{vnr}}}$	Duration of virtual network request n_{vnr}
$N^v_{n_{\text{vnr}}}$	Virtual node set for virtual network requests n_{vnr}
$b_{S,fm(v_2)}$	A function call that returns the hop distance between two physical nodes
$B_{S,fm(v_2)}$	A calling function that returns the number of bits that can be encoded per symbol, determined by the distance between physical nodes
V	Virtual machine collection
n_{vnr}	Map the successful set of virtual network requests
v	Because VMs are mapped to virtual nodes, VMs and virtual nodes can be represented

3　基于虚拟拓扑图的虚拟机放置算法设计

Rugwiro等提出了基于K-Means的虚拟机放置算法，其算法的步骤如图1所示。图1(a)虚拟请求有8个待放置的虚拟机，圆圈内的数字表示虚拟机的名称，记录的是第几个虚拟机，虚拟机链接上的数字表示虚拟机间业务量的大小。从图1(b)可以看出，虚拟请求被分为5簇，分别为P1、P2、P3、P4、P5，其中簇P1包含4个虚拟机，其余的簇里仅有一个虚拟机。图1(c)表示将分簇后的虚拟机放置到数据中心胖树拓扑中物理服务器的结果，S1～S20是物理交换机，S21～S36是物理服务器。从图1(c)可以看出，上述的分簇算法使用的物理网络带宽资源为67×2+48×4+17×4+11×6=460。

图2表示本文提出的虚拟机放置算法，在基于K-Means分簇的情况下将虚拟机v_4、v_7、v_6、v_1分为一簇，其余的虚拟机分为另一簇，此放置方法解决了K-Means算法中簇大小不一的问题。从图2可以看出，分簇方法使用的物理网络带宽资源为67×2+48×2+17×2+11×2=286，大大节省了网络中带宽资源。

3.1　资源约束

本文的底层物理网络是数据中心与光网络融合的Fat-Tree物理网络，因此在考虑虚拟机放置问题约束时，要考虑两阶段的约束(节点映射阶段约束、链路映射阶段的约束)。

1)节点映射阶段约束条件

$$\sum_{S_i} M^{v_j}_{s_i}=1,\quad \forall S_i。 \tag{6}$$

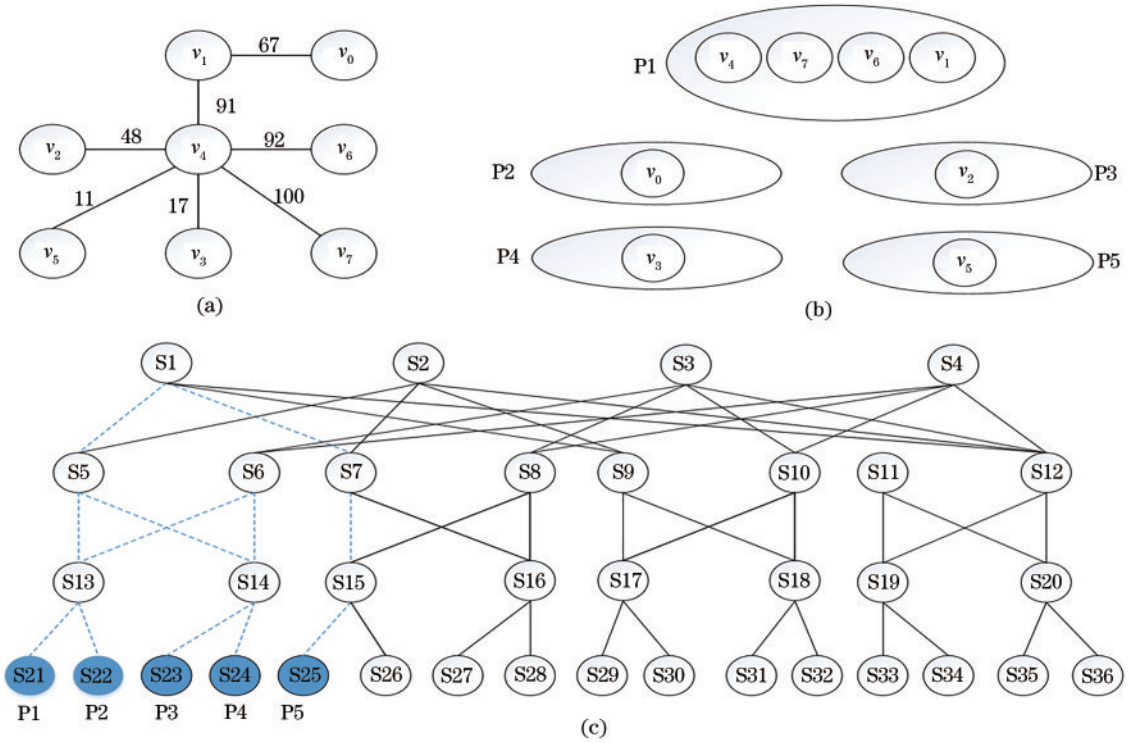

图 1 *K*-Means 虚拟机算法放置。(a)软管模型转化后的管道模型虚拟数据中心请求;(b) *K*-Means算法分簇的结果;(c)虚拟机放置到数据中心胖树拓扑中物理服务器

Fig. 1 Placement of *K*-Means virtual machine algorithm. (a) Pipeline model virtual data center request after hose model transformation; (b) clustering result of *K*-means algorithm; (c) virtual machine placement to physical server in data center fat-tree topology

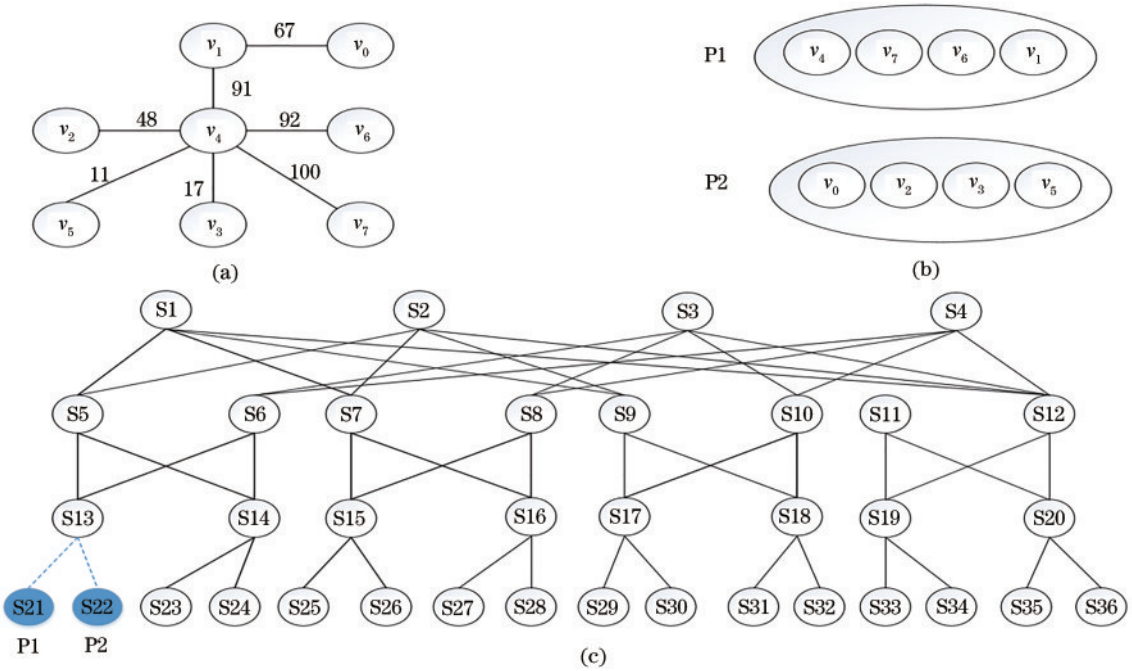

图 2 VT-VMPA 虚拟机算法放置。(a)软管模型转化后的管道模型虚拟数据中心请求;(b) VT-VMPA算法分簇的结果;(c)虚拟机放置到数据中心胖树拓扑中物理服务器

Fig. 2 Placement of the VT-VMPA algorithm. (a) Pipe model virtual data center request after the hose model transformation; (b) result of VT-VMPA algorithm clustering; (c) virtual machine is placed to the physical server in the data center fat-tree topology

式(6)表示在映射虚拟数据中心请求时，对于任意一个虚拟机的请求只能映射到一个物理服务器。

$$\begin{cases}\sum_{j=1}^{k} HR_{\mathrm{CPU},j}\times M_{s_i}^{v_j}\leqslant P_{\mathrm{CPU},i},\ \forall i\\ \sum_{j}^{k} M_{s_i}^{v_j}\leqslant U_{\max},\ \forall i\end{cases}。\tag{7}$$

式(7)中：上式表示放置在物理服务器上的总的虚拟机所请求的CPU资源不应该超过该物理服务器的CPU计算容量；下式表示放置在物理服务器上的虚拟机不应该超过$U_{\max}$，$U_{\max}$设置为虚拟数据中心请求中虚拟机数量的一半。

2) 链路映射阶段约束条件

一个M台虚拟机的虚拟数据中心请求，该请求中放置在服务器S_i上的虚拟机集合为Vs_i，放在S_k上的虚拟机集合为Vs_k，故链路映射阶段的带宽约束表示为

$$\sum_{x\in Vs_i}\sum_{y\in Vs_k} f_{x,y}\leqslant B_{i,k}。\tag{8}$$

3) 权重条件

$$W_{\mathrm{weight},\,S}=\begin{cases}\dfrac{R_{\mathrm{CPU},v}}{R_{\mathrm{CPU},S}}, & v\ \text{is the first mapped virtual node}\\ \dfrac{R_{\mathrm{CPU},v}}{R_{\mathrm{CPU},S}}+\dfrac{\sum_{v_2\in\phi_v} e_{v,v_2}b_{S,fm(v_2)}}{\sum_{v_2\in\phi_v} e_{v,v_2}B_{S,fm(v_2)}}, & \text{other}\end{cases},\quad \forall S\in\varphi_v,\tag{9}$$

式中：$R_{\mathrm{CPU},v}$是虚拟节点v所需的CPU计算量；$R_{\mathrm{CPU},S}$是数据中心节点的可用CPU计算量；ϕ_v是已经映射的虚拟节点集；e_{v,v_2}是一个布尔变量，为1时表示虚拟节点v、v_2之间有边；$b_{S,fm(v_2)}$是一个函数调用，返回的是两物理节点间的跳距；$B_{S,fm(v_2)}$也是一个调用函数，返回的是由物理节点间的距离决定的每个符号可编码的比特数。

3.2 虚拟机映射阶段

基于虚拟拓扑的虚拟机放置算法首先将虚拟机集合划分为一系列不重叠的子集合，然后将每一个子集合虚拟为一个超级节点，最后为每一个超级节点寻找一台物理服务器。可见虚拟机放置算法的关键是如何划分不重叠的子集合，而本文虚拟机放置问题，考虑的优化目标是减少物理网络资源的使用，因此将虚拟机间的业务量和虚拟的超级节点的数量作为划分子集合的标准。

给定由M台虚拟机组成的虚拟数据中心请求$v=\{v_0,v_1,\cdots,v_{M-1}\}$，先根据虚拟机间的业务量需求矩阵，计算每一个虚拟机v_j与其他虚拟机间的总业务量，

$$f_{j,\mathrm{sum}}=\sum_{k=0}^{M-1} f_{j,k},\quad j\neq k。\tag{10}$$

在划分请求第一个子集合的过程中，首先要选出第一个子集合的核心虚拟机，选择标准为

$$V_{\mathrm{core}}=\arg-\max f_{j,\mathrm{sum}},\quad 0\leqslant j\leqslant M-1,\tag{11}$$

$$V_{\min}=\arg-\min f_{v_j,\mathrm{sum}},\quad v_j\in P*。\tag{12}$$

选取子集合的步骤如下。

步骤1：确定V_{core}，构成当前超级节点Super$V=\{V_{\mathrm{core}}\}$，将$P*$为空

步骤2：将剩下的虚拟机集合$V=V-V_{\mathrm{core}}$，再为核心虚拟机寻找合适物理服务器$S_{V_{\mathrm{core}}}$

步骤3：将与V_{core}相连但没有映射的虚拟机按照与V_{core}通信量降序得到VM_{sorted}

步骤4：将VM_{sorted}中虚拟机合并到超级节点Super$V$，确保超级节点的总CPU资源需求小于服务器$S_{V_{\mathrm{core}}}$的剩余CPU计算容量、总的虚拟机数量小于$U_{\max}$

步骤5：将剩余未映射的虚拟机放入$P*$，将$P*$作为当前超级节点，为其寻找物理服务器S_{P*}

步骤6：判断$P*$能否放入服务器S_{P*}中，若能则虚拟机集合放置完毕。不能则需要不断从$P*$中选择对外业务量最小的$V_{\min}$放入$P**$中，直到$P*$能整体放入S_{P*}中

步骤7：将$P**$赋值给$P*$，重复步骤5和6，直到所有虚拟机都完成映射

虚拟机映射阶段算法伪代码如下。

1: for 虚拟数据中心请求中的虚拟机集合V do

2: 根据虚拟机集合中虚拟机与其他虚拟机的业务量，降序排序形成虚拟机集合VM

3: end for

4: 从集合VM中选取核心虚拟机V_{core}，构成超级节点Super$V=\{V_{\mathrm{core}}\}$，则剩余的虚拟机集$VM=VM-V_{\mathrm{core}}$

5: 找出Fat-Tree数据中心中满足V_{core}CPU计算资源的物理节点，形成候选的服务器$S_{\mathrm{select}}^{V_{\mathrm{core}}}$

6: for 每一个物理服务器$S_i\in S_{\mathrm{select}}^{V_{\mathrm{core}}}$ do

7: 计算权重$W_{\mathrm{weight},\,S_i}$，并将服务器和服务器对应的权重以键值对的形式存入字典

8: end for

9: 从字典中选择权值最小的物理服务器$S_{\min}$用来承载核心虚拟机V_{core}

10: 将已使用的物理服务器$S_{\min}$添加到集合Ψ中，将已映射的虚拟机V_{core}加入Ψ集合中

11:获取虚拟数据中心请求第一个子集合可合并的最大虚拟机数 U_{max} 以及当前服务器消耗的CPU资源量 CpuTotal(V_{core})
12:将虚拟请求中与 V_{core} 相连但未做映射的虚拟机按照与 V_{core} 的通信量降序得到 VM_{sorted}
13:for 每一个虚拟机 $v_j \in VM_{sorted}$ do
14: if R_{CPU,V_j} + CpuTotal(V_{core}) < $P_{CPU,S_{min}}$ and len (Super$V$) < U_{max} then
15: 将该虚拟机映射到物理服务器 S_{min},修改 CpuTotal(V_{core}) = CpuTotal(V_{core}) + R_{CPU,v_j},在集合 VM 中删除 v_j,Super$V$ = Super$V \cup v_j$,将虚拟机 v_j 加入到 Ψ。
16: else
17: 退出第一个子集合中虚拟机的合并过程
18:P^* 等于集合 VM 中剩余的虚拟机
19:while len(P^*) ≠ 0 do
20: 置为 P^{**} 空
21: for 每一个虚拟机 $v_k \in P^*$ do
22: 找出物理网络中满足 v_kCPU计算资源的物理节点,形成候选的服务器 $S_{select}^{v_k}$
23: for 每一个物理服务器 $S_k \in S_{select}^{v_k}$ do
24: 计算权重 $W_{weight,\ s_i}$,存入字典中
25: end for
26: 从字典中选择权值最小的物理服务器 S_{min}^* 用来承载虚拟机 v_k
27: 将使用的物理服务器 S_{min}^* 添加到集合 Ψ,将已映射的虚拟机 v_k 加入集合 Ψ
28: 获取集合 P^* 的CPU资源消耗 CpuTotal(P^*)
29: if CpuTotal(P^*) < P_{CPU,S_{min}^*} then
30: 将 P^* 中的每一个虚拟机映射到服务器 S_{min}^* 中,并将其加入到集合 Ψ 退出虚拟机映射
31: else
32: while CpuTotal(P^*) > P_{CPU,S_{min}^*} do
33: 从 P^* 中删除对外务量最小的 V_{min},并将其加入到集合 P^{**} 中
34: for 每一个虚拟机 $v_x \in P^*$ do
35: 将 P^* 中的每一个虚拟机映射到服务器 S_{min}^* 中,将其加入到集合 Ψ
36: 退出本次子集合中虚拟机的合并过程
37: $P^* = P^{**}$
38:if 虚拟数据中心请求中的所有虚拟机都映射成功 then
39: return 映射成功,记录映射关系
40:else
41:return 映射失败

3.3 链路映射阶段

链路映射阶段的主要思想是:将虚拟机间的业务关系转化为簇间的业务关系,并按照簇间业务量大小降序存入虚拟链路集合中。然后对于簇间的每一个业务量,根据虚拟机映射阶段的结果找到物理网络中对应的服务器,再采用KSP算法计算两服务器间的 K 条最短路,选择连续频谱隙资源最多的路径承载该业务量并判断路径中的每一条链路上是否有满足要求的带宽资源分配给此虚拟链路,若没有,则请求映射失败,若有,选择合适的调制格式为业务分配带宽资源,直到所有的簇间业务都已经映射成功,则请求映射成功。其伪代码如下。

1: 将每一个子集合虚拟为超级节点,并根据管道模型虚拟数据中心的虚拟机间的业务关系转化为超级节点间的业务关系,存入字典中
2: 将字典中的元素按照超级节点间业务量的大小降序排序
3: for超级节点间的每一个业务量do
4: 根据虚拟机映射阶段的结果找到Fat-Tree数据中心网络中对应的服务器,采用KSP算法计算两服务器间的K条最短路
5: 从K条最短路中选择连续频谱隙资源最多的路径并判断其是否满足超级节点间的业务量所需的带宽资源,若满足,则将该路径赋值给最终选择的路径SelectPc,链路映射成功,否则链路失败
6: end for
7: if超级节点间的所有业务量映射成功 then
8: return虚拟数据中心映射成功,分配物理网络资源,记录虚拟数据中心
9:else
10:return 虚拟数据中心映射失败

3.4 与其他分簇方式的虚拟机放置算法对比分析

根据聚类分簇算法中对象间相似度的计算方式以及聚类结果中对象的关系可以将聚类分簇算法分为划分法、层次法、基于密度的方法、基于网格的方法。

根据文献[18]所描述可知,基于划分的聚类方法只是将数据对象集合划分为若干个无交集的子集(簇),使得每个对象仅属于一个子集。基于层次聚类把数据对象构建成一组具有树状结构的嵌套簇,除了叶子节点的每个簇都是由其子节点(子簇)的并集构成,根节点包含所有的数据对象。基于密度的聚类方法是根据数据集密度的大小将数据集分类成簇,密度高的区域聚类成簇,密度低的区域作为噪声和孤立点处理。基于网格的聚类方法把原数据对象空间划分成独立于输入对象分布的单元。通过构建父级子级网格单元关系形成一种多分辨率的网络数据结构,将连续空间离散化成有限数目的单元,用所形成的网格结构进行聚类。

基于划分的聚类方法主要是 K-Means算法,本文主要讨论的是与这种方法的区别。基于层次聚类方法主要是单链接聚类、CURE算法和BIRCH算法,与本文算法相比,基于层次聚类分簇方法会产生一个较大的簇,再将这个较大的簇放置在服务器上时会导致服务器CPU计算资源快速消耗,网络局部瘫痪。基于密度的分类方法主要是DBSCAN算法,该算法与VT-VMPA算法相比,它产生有的簇虚拟机的数量过多,有的簇虚拟机的数量过少,甚至一个虚拟机也可以单独作为一个簇问题,导致开启的服务器的数量增多,还

会造成簇间的业务量占用网络中过多的带宽资源。基于网格的聚类分簇方法主要是STING算法，与本文算法相比，STING算法依赖密度阈值的选择，密度太高会导致簇丢失而密度太低会导致本应该分开的簇合并成一个大簇，从而影响网络中的带宽资源。

4 仿真实验及分析

在数据中心与光网络融合的Fat-Tree物理网络中，每台服务器提供的CPU计算资源为500，服务器与交换机以及交换机与交换机之间的频谱栅格数均为320，每一个频谱隙的带宽资源为12.5 GHz。虚拟数据中心请求由python程序随机生成，虚拟机的数量在2～10之间，每个虚拟机所需的CPU计算资源为1～10之间的随机数，虚拟机间的业务以0.5的概率产生，业务量大小为10～100之间的随机数。虚拟数据中心请求动态的到达和离去，通过python程序控制，请求总数为500。

本文设计了在网络业务强度 $T_{\text{Intensity}}$（Erlang）固定和虚拟机数量 M 固定的两种场景下，从平均带宽消耗、阻塞率和平均收益来衡量基于跳距和距离自适应的虚拟机放置算法（HDA-VMPA）、改进的基于簇的虚拟机放置算法（IC-VMPA）和VT-VMPA算法的优劣。但是由于实验条件所限，尤其是缺乏搭建灵活栅格网络的设备和仪器，难以增加原理性实验，故本文采用仿真实验的方法分析比较了所提算法的性能。

4.1 固定网络业务强度下性能对比

本文将业务强度固定在80 Erlang，通过增加虚拟机的个数，来比较三个算法的优劣。

由图3可知，伴随着虚拟机数量增加，三种算法的平均带宽消耗都在增加，但是VT-VMPA算法消耗的平均带宽更少。这是因为：与HDA-VMPA算法相比，VT-VMPA算法可以将同一个虚拟数据中心请求中的不同虚拟机放置到同一个服务器下，大大减少物理网络中的资源消耗；与IC-VMPA算法相比，该算法在保证服务器CPU计算资源不急速消耗的前提下，解决了IC-VMPA算法中分簇后簇间的业务量占用过多网络资源的问题，减少了物理网络资源消耗。

从图4可以看出，虚拟机数量增加，会导致3种算法的阻塞率都增加，这是由于Fat-Tree物理网络中的资源是有限的，而随着虚拟数据中心请求数量的增加和虚拟数据中心请求中虚拟机数的增加，算法通过拒绝不满足约束条件的VDC请求来给已经完成映射的VDC请求提供服务质量保证。VT-VMPA算法的阻塞率是最低的，因为相比于HDA-VMPA算法，IC-VMPA算法和VT-VMPA算法可以将多台虚拟机虚拟为一个超级节点作为整体放入到一个物理服务器下，减少了物理网络中的资源消耗，提高了映射率，降低阻塞率。此外，VT-VMPA算法的阻塞率低于IC-VMPA算法的原因在于前者不仅考虑了虚拟机间的连接关系对分簇的影响，还考虑到簇的数量对映射结果的影响，尽可能地减少簇间业务量对网络资源的消耗。

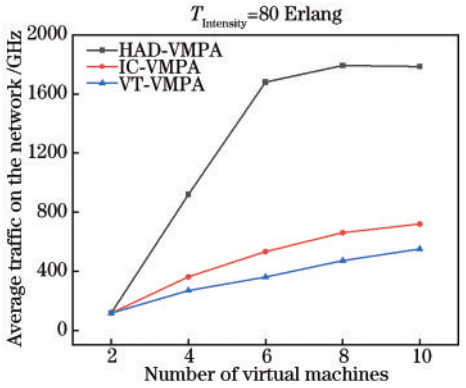

图3 三种算法的网络平均带宽消耗

Fig. 3 Average network bandwidth consumption of the three algorithms

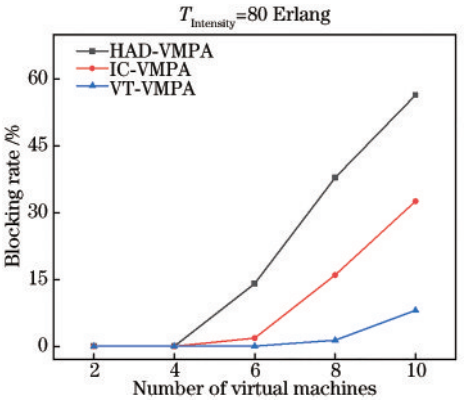

图4 不同数量虚拟机下三种算法的阻塞率

Fig. 4 Blocking rate of three algorithms with different number of virtual machines

从图5可以看出，当虚拟机数量从2增加到6时，三种算法的平均收益也在增加，这是因为虚拟请求规模较大时有较多的收入，在请求阻塞率上升不明显的情况下，会增加时间平均收益。但当虚拟机数量从6增加到10时，HDA-VMPA算法阻塞率的剧增，导致

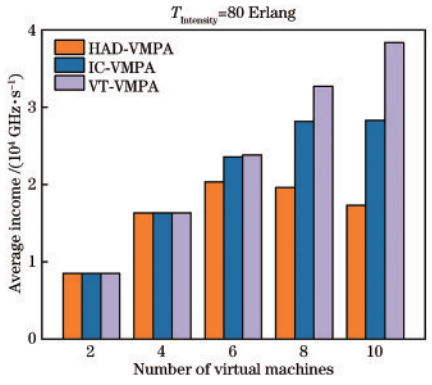

图5 不同数量虚拟机下三种算法平均收益

Fig. 5 Average revenue of the three algorithms under different number of virtual machines

其时间平均收益相对虚拟机数量为6时减少近15%。在任何规模虚拟数据中心请求下，VT-VMPA算法相比于其他两种算法具有更高的时间平均收入或者说至少保持持平状态。

4.2　固定虚拟机数量下性能对比

本文将虚拟机的数量固定在8台，通过增加业务强度来比较三种算法的性能优劣。

由图6可知，三种算法的阻塞率都随着网络业务强度的增加而上升，这是因为数据中心网络同时映射了更多的虚拟数据中心请求，物理资源变少，从而拒绝后续到达的不满足资源约束的请求，导致阻塞率上升。而在任何流量负载下，VT-VMPA算法的阻塞率都低于其他两种算法，这是因为该算法不仅能将不同的虚拟机作为整体映射到同一个物理服务器下进而减少不同虚拟机间的业务量对网络资源的占用，还松弛了虚拟拓扑的连接关系对分簇后簇的大小的影响，减少了簇间业务量对网络资源的消耗进而降低了阻塞率。另外，随着网络中业务强度的增加，三种算法阻塞率的差值大体上呈上升趋势，这是因为：虚拟机数量较少时，虚拟机间的业务数量较少，物理资源充足，所以阻塞率差值不大；当虚拟机数量较多时，HDA-VMPA算法、IC-VMPA算法的簇间业务数量剧增，物理网络链路带宽资源使用较为严峻，难以接受后续到达的虚拟数据中心请求，但VT-VMPA算法将较多的虚拟机放入一个物理服务器中，虚拟机间的业务量较多地转化为服务器内部的通信，不会过多地占用物理网络的资源，节省大量的带宽资源，所以算法间阻塞率的差值较大。

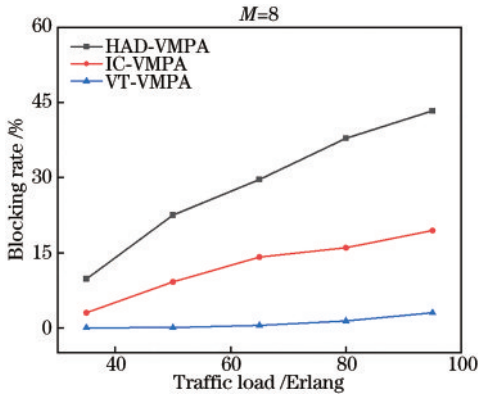

图6　不同业务强度下三种算法阻塞率

Fig. 6　Blocking rate of three algorithms under different service intensities

从图7中可以很明显地看出，因为流量负载从10 Erlang增加到50 Erlang，可以整体地看出三种算法网络中平均带宽消耗的情况，若流量负载从35 Erlang开始分析，则HDA-VMPA算法的网络平均带宽资源消耗已达到1400 GHz，之后随着流量负载的增加缓慢增加。如图所示，VT-VMPA算法对网络资源的占用

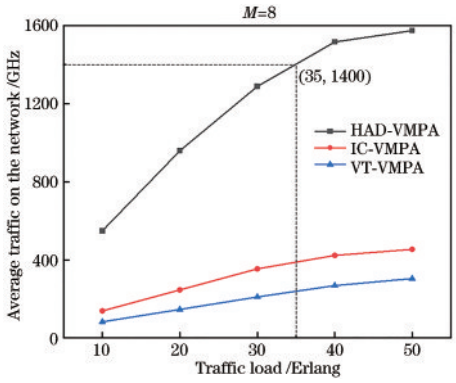

图7　不同业务强度下算法的网络平均带宽消耗

Fig. 7　Average network bandwidth consumption of algorithms under different service intensities

在所有情况下都小于HDA-VMPA算法与IC-VMPA算法，其降低的幅度分别约84%和38%。这主要由于VT-VMPA算法在保证资源不急速消耗的前提下将虚拟机分为更少的簇，从而大大降低了簇间业务量对网络资源的消耗。

从图8可以看出，随着流量负载的增加，三种算法的平均收益也都随之增加，与HDA-VMPA算法和IC-VMPA算法相比，VT-VMPA算法的收益增加明显，增加了近117%，这是因为在虚拟数据中心请求规模相同的情况下，VT-VMPA算法的阻塞率最小（为3%），接受的虚拟数据中心请求最多，而接受的请求越多，收益就越多。

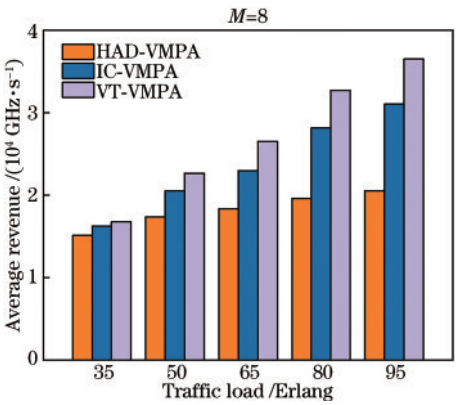

图8　不同业务强度下三种算法的时间收益

Fig. 8　Time revenue of the three algorithms under different service intensities

5　结　　论

针对弹性光数据中心承载软管虚拟数据中心时簇间业务量占用过多网络资源的问题，提出了VT-VMPA算法。该算法通过虚拟机集合划分为一系列不重叠的子集合，然后将每一个子集合虚拟为一个超级节点，最后为每一个超级节点寻找一台物理服务器，通过这种方法来减少网络资源使用。仿真结果表明，与其他算法相比，该算法能够减少物理资源带宽的消

耗、减小阻塞率、提高映射成功率与时间平均收益。

但该算法也有不足之处，因为该算法只能在一定程度上减少对网络资源的占用，没有最大化减少分簇后簇间的业务量从而最大程度上节约网络带宽资源。同时，VT-VMPA算法只考虑虚拟机间的业务量和虚拟的超级节点的数量，并以此作为子集合的标准，没有考虑到虚拟机之间连接的传输时间，因此会导致虚拟机之间传输时延变长。本文在现在研究中采用灵活栅格光网技术来实现传统胖树结构的数据中心，即架顶交换机采用胖树结构互联，提出了软管模型虚拟机部署时在该数据中心的映射模型和放置算法，下一步工作考虑采用灵活栅格技术实现架顶交换机之间更少跳数，甚至直达互联的光数据中心架构，以及该架构上的虚拟机部署算法和性能评估。

参 考 文 献

[1] Duffield N G, Goyal P, Greenberg A, et al. A flexible model for resource management in virtual private networks[J]. ACM SIGCOMM Computer Communication Review, 1999, 29(4): 95-108.

[2] Lu P, Zhang L, Liu X H, et al. Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks[J]. IEEE Network, 2015, 29(5): 36-42.

[3] 朱伟, 李晶, 裴丽, 等. 基于偏振延时干涉的瞬时频率测量系统的分析与优化[J]. 光学学报, 2021, 41(21): 2107001.
Zhu W, Li J, Pei L, et al. Analysis and optimization of instantaneous frequency measurement system based on polarization time delay interference[J]. Acta Optica Sinica, 2021, 41(21): 2107001.

[4] 刘元, 李晶, 贺永娇, 等. 基于双平行马赫-曾德尔调制器和平衡光电探测器的四倍频可调对称三角形函数波形信号发生器[J]. 光学学报, 2021, 41(19): 1906005.
Liu Y, Li J, He Y J, et al. Generator of signals with quadruple frequency and triangular waveform tunable in symmetry based on dual-parallel Mach-Zehnder modulator and balanced photodetector[J]. Acta Optica Sinica, 2021, 41(19): 1906005.

[5] Zhao J Z, Subramaniam S. Virtual network mapping in elastic optical networks with sliceable transponders[J]. Photonic Network Communications, 2020, 40(9): 281-292.

[6] Feng T, Bi J, Hu H Y, et al. Networking as a service: a cloud-based network architecture[J]. Journal of Networks, 2011, 6(7): 1084-1090.

[7] Sahoo P K, Dehury C K, Veeravalli B. LVRM: on the design of efficient link based virtual resource management algorithm for cloud platforms[J]. IEEE Transactions on Parallel and Distributed Systems, 2018, 29(4): 887-900.

[8] Qin Y, Wang H, Yi S W, et al. Virtual machine placement based on multi-objective reinforcement learning [J]. Applied Intelligence, 2020, 50(8): 2370-2383.

[9] Chen M T, Hsu C C, Kuo M S, et al. GreenGlue: power optimization for data centers through resource-guaranteed VM placement[C]//2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing, September 1-3, 2014, Taipei, Taiwan, China. New York: IEEE Press, 2014: 510-517.

[10] Luo G Y, Qian Z Z, Dong M X, et al. Improving performance by network-aware virtual machine clustering and consolidation[J]. The Journal of Supercomputing, 2018, 74(11): 5846-5864.

[11] Zhang B B, Wang X, Wang H. Virtual machine placement strategy using cluster-based genetic algorithm [J]. Neurocomputing, 2021, 428: 310-316.

[12] Alnoman A. Machine learning-based task clustering for enhanced virtual machine utilization in edge computing [C]//2020 IEEE Canadian Conference on Electrical and Computer Engineering, August 30-September 2, 2020, London, ON, Canada. New York: IEEE Press, 2020.

[13] Wei C, Hu Z H, Wang Y G. Exact algorithms for energy-efficient virtual machine placement in data centers [J]. Future Generation Computer Systems, 2020, 106: 77-91.

[14] Li R, Zheng Q H, Li X Q, et al. Multi-objective optimization for rebalancing virtual machine placement[J]. Future Generation Computer Systems, 2020, 105: 824-842.

[15] Sadegh S, Zamanifar K, Kasprzak P, et al. A two-phase virtual machine placement policy for data-intensive applications in cloud[J]. Journal of Network and Computer Applications, 2021, 180: 103025.

[16] Rugwiro U, Gu C H. Customization of virtual machine allocation policy using k-means clustering algorithm to minimize power consumption in data centers[C]//ICC '17: Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing, March 22-23, 2017, Cambridge, United Kingdom. New York: ACM Press, 2017: 1-8.

[17] Liu F Q, He W G. Hose-model network virtualization in flexgrid optical networks[J]. Journal of Optical Communications and Networking, 2020, 12(3): 13-23.

[18] 王玉晗, 罗邓三郎. 聚类算法综述[J]. 科技资讯, 2018, 16(24): 10-11.
Wang Y H, Luo D S L. Overview of clustering algorithms[J]. Science & Technology Information, 2018, 16(24): 10-11.