计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20151134
53(4):785-797, 2016

软件定义数据中心内一种基于拓扑感知的 VDC 映射算法

文学敏[1] 韩言妮[1] 于 冰[1] 孙建朋[2] 徐 震[1]
[1](信息安全国家重点实验室(中国科学院信息工程研究所) 北京 100093)
[2](山东大学信息科学与工程学院 济南 250100)
(wenxuemin@iie.ac.cn)

A Topology-Aware VDC Embedding Algorithm in Software-Defined Datacenter

Wen Xuemin[1], Han Yanni[1], Yu Bing[1], Sun Jianpeng[2], and Xu Zhen[1]
[1](*State Key Laboratory of Information Security* (*Institute of Information Engineering*, *Chinese Academy of Sciences*), *Beijing* 100093)
[2](*School of Information Science and Engineering*, *Shandong University*, *Jinan* 250100)

Abstract In cloud computing environment, service provider (SP) can pay for the resources from infrastructure provider (InP) on-demand to deploy their services. In the case, SP can focus on service business without considering their physical infrastructures and expertise of maintenance. Only providing resources in term of virtual machines, the traditional InPs do not ensure network performance and bandwidth isolation. As the network virtualization is developed, especially the SDN concept, some researchers advocate InPs to provide resources in term of virtual data center (VDC) to solve these limits. Despite many advantages of VDC, there is also a new challenge that is the VDC embedding problem known as an NP-hard problem. With the goal of minimal cost and maximal revenue, it solves the problem of allocating resources to fulfill the SPs' requirements. Considering the tradeoff of VDC reliability and embedding cost, a VDC embedding algorithm based on topological potential and modularity is proposed to improve acceptance ratio and the InPs' revenue. Moreover, we further optimize the algorithm based on a given threshold by selecting high Revenue/Cost ratio VDCs. Extensive simulations show that compared with the existing algorithms, our approach is capable of reducing the core bandwidth consumption in data center. Furthermore, these proposals can accept more VDCs and obtain more revenue.

Key words cloud computing; data center; software defined networking (SDN); network virtualization; virtual data center (VDC) embedding

摘 要 在云计算中,服务提供商(service provider, SP)可以向基础设施提供商(infrastructure provider, InP)按需租赁资源并部署服务.SP 只需专注于自己的服务即可,无需考虑设备成本与维护代价.然而传统 InP 仅以虚拟机的方式提供资源,并不保证网络性能与带宽隔离.随着网络虚拟化技术的发展,尤其是软件定义网络(software defined networking, SDN)概念的提出,一些研究人员建议 InP 以虚拟数据中心(virtual data center, VDC)的方式为 SP 提供资源,以解决传统数据中心的上述问题.尽管以 VDC 的方式分配资源具有诸多的优势,也带来了一项新的挑战,如何满足 SP 的多样化需求,以最

收稿日期:2015-12-21;**修回日期**:2016-02-03
基金项目:国家自然科学基金项目(61303252,61303250);中国科学院战略性先导科技专项基金项目(XDA06010300)
This work was supported by the National Natural Science Foundation of China (61303252, 61303250) and the Strategic Priority Research Program of the Chinese Academy of Sciences (XDA06010300).
通信作者:韩言妮(hanyanni@iie.ac.cn)

小的代价、最大的收益为 VDC 分配资源,这是一个 NP-hard 问题.为解决 VDC 映射问题,提出了一种基于拓扑势和模块度的启发式映射算法,折衷租户的可靠性需求与映射代价,并提高 InP 收益.最后,基于收益代价比门限经验值,提出一种动态监控策略,选择高收益代价比的 VDC 请求,进一步最大化 InP 的利润.大量的仿真实验证明该算法可以以最小的代价接受更多的请求,同时提高 InP 收益.

关键词 云计算;数据中心;软件定义网络;网络虚拟化;VDC 映射

中图法分类号 TP393.01

随着科学、商业等各方面的信息化水平不断提升,各行各业与互联网产业不断碰撞与融合.在“互联网+”的不断发展进程中,信息化社会所产生的数据也日益庞大.在人们对数据的存储、计算、传输提出更高需求的背景下,云计算作为一种变革式的计算模型被提了出来.作为下一代计算模式,云计算将成为“互联网+”社会的信息中枢,它允许用户按需付费,随时随地使用远程的计算、网络、存储资源.在云计算中,基础设施提供者(infrastructure provider, InP)将基础网络设施资源抽象化为池.服务提供者(service provider, SP)向 InP 租赁一定的资源就可以部署自己的服务或计算任务.InP 可以实现统一的资源管理与调度,提高物理资源的利用率与收益;SP 只需专注于业务逻辑,无需投入大量基础设施成本以及维护知识和代价.作为一种高效的计算模型,云计算吸引着越来越多的企业和科研机构的关注.

然而,当前的 InP,比如亚马逊 EC2,主要以虚拟机(VM)的形式为用户提供资源,并不提供网络隔离和带宽保障.建立在 TCP/IP 协议栈之上的网络仅提供尽力而为的服务模式.这种传统服务模式在网络隔离、安全、网络性能保障、自动化管理等诸多方面存在着问题[1].为了解决传统数据中心的这些问题,研究人员建议 InP 以虚拟数据中心(virtual data center, VDC)的形式为租户分配资源[1-2].所谓虚拟数据中心是指建立在物理网络切片上的虚拟资源集合,包括虚拟机、虚拟交换机、虚拟路由以及连接它们的虚拟链路[2].相比于传统仅提供 VM 的方式,以 VDC 的方式分配资源可以做到网络隔离和带宽保障.

VDC 的实现依赖于网络虚拟化技术.与发展快速且成熟的主机虚拟化技术相比,网络虚拟化发展相对滞后.软件定义网络(software defined networking, SDN)概念的出现和发展,加速了网络虚拟化的进程.简单来说,SDN 将控制平面从数据转发平面分离出来,可以通过控制器来协同转发组件的行为、获取流量统计数据以及监控拓扑变化[3].SDN 具有集中式控制、全网信息获取和网络功能虚拟化等特性,利用这些特性可以有效解决数据中心出现的各种问题[4].当前数据中心内 SDN 应用的部署得到极大的发展,其中性能和节能是重点考虑的 2 个方面[4-5].基于 SDN 的网络虚拟化技术可以有效方便地在数据中心物理网络之上建立 VDC,可以帮助 InP 以 VDC 的形式出租资源,并为多租户之间提供网络隔离与带宽保障.由于多个 VDC 在逻辑上是分离的,SP 可以完全控制自己所租赁的 VDC,比如引入自定义的网络协议或流量策略.

尽管以 VDC 的形式提供资源可以解决数据中心中的性能、安全等问题,但同时也带来了一项新的挑战,这就是 VDC 映射问题.所谓 VDC 映射问题,是指在数据中心内如何灵活、高效地为 VDC 请求分配合适的物理资源.一个优秀的 VDC 映射算法可以帮助 InP 达到多种优化目标,比如最大化资源利用率、最大化收益、最小化维护代价.VDC 映射问题虽然与被广泛研究的虚拟网(VN)映射问题[6]很相似,但还是存在着一些区别:1)VN 映射问题主要面向于广域网,而 VDC 映射问题主要面向于数据中心内的资源分配;2)VN 中的节点考虑的是广域网中的转发设备,而在 VDC 中可以包含多种节点,比如主机、路由、交换机、存储节点等[1];3)对同一个请求,VN 映射问题中一个物理节点只能映射一个虚拟节点[6],然而在云计算环境中同一 VDC 的多台 VM 可以分配到一台物理主机中.

本文主要探讨了软件定义数据中心中的 VDC 映射问题.主要贡献如下:1)提出了基于 SDN 的数据中心资源管理框架;2)对 VDC 映射问题进行了建模,并分析了 VDC 映射过程中影响 InP 收益的因素;3)考虑多种资源需求(CPU、内存、带宽)以及 VDC 可靠性需求,提出了一种新的资源分配算法;4)通过大量的仿真实验验证该算法能够以较低的代价获取较高的收益,最后提出基于一种动态监控策略选择高收益代价比的 VDC 请求,进一步优化算法.

1 问题描述

1.1 VDC 映射问题与数据中心资源管理框架

InP 拥有物理基础设施,通过向 SP 出租资源获取收益. SP 需要一定的基础设施部署自己的服务或任务. 为了节约设备成本和维护代价,SP 可以向 InP 以 VDC 的形式租赁资源. 首先,当 SP 需要租赁资源时,它需要向 InP 提交 VDC 请求说明,包括 VDC 的拓扑、每个虚拟节点的 vCPU 数量和内存需求量、每条虚拟链路的最小带宽. InP 收到请求后,根据底层网络的状态,为 VDC 分配满足需求的物理资源. InP 会在不同的时间收到来自不同 SP 的不同 VDC 请求. 但 InP 无法预知 VDC 请求的到达时间,也无法预知 VDC 请求的具体内容. 所以,本文所考虑的 VDC 映射问题是一个在线映射问题.

InP 可以在底层网络之上建立多个网络切片,并为不同网络切片提供隔离. 每个网络切片拥有自己的 IP 地址空间,互不干涉. 从 SP 的角度看,它可以像控制真实网络一样控制自己的网络切片,底层网络对它来说是透明的. SP 不仅仅可以直接使用虚拟网络切片,它也可以通过自己的 SDN 控制器来控制虚拟网络的转发行为. 本文提出了一个基于 SDN 的数据中心资源管理框架,如图 1 所示. 类似于传统的主机操作系统,框架分为 5 层:基础物理网络设施、控制接口层、网络操作系统(NOS)、系统控制层和用户控制层. 下面分别介绍各层的功能特点:

1) 基础物理网络设施. 该层是 InP 所拥有的底层网络设施. 本文主要考虑的是单个数据中心网络,而实际中既可以是单个数据中心,也可以是通过骨干网互联的多个数据中心.

2) 控制接口层. 该层主要由 SDN 控制接口和主机控制接口组成,为上层网络操作系统提供接口. SDN 控制接口可以控制物理交换机和虚拟交换机的转发行为、查询交换机中的流表信息、统计信息以及监控物理网络拓扑变化. 主机控制接口可以控制虚拟机的创建、删除、修改和迁移、查询物理主机、虚拟机的使用状态和统计数据.

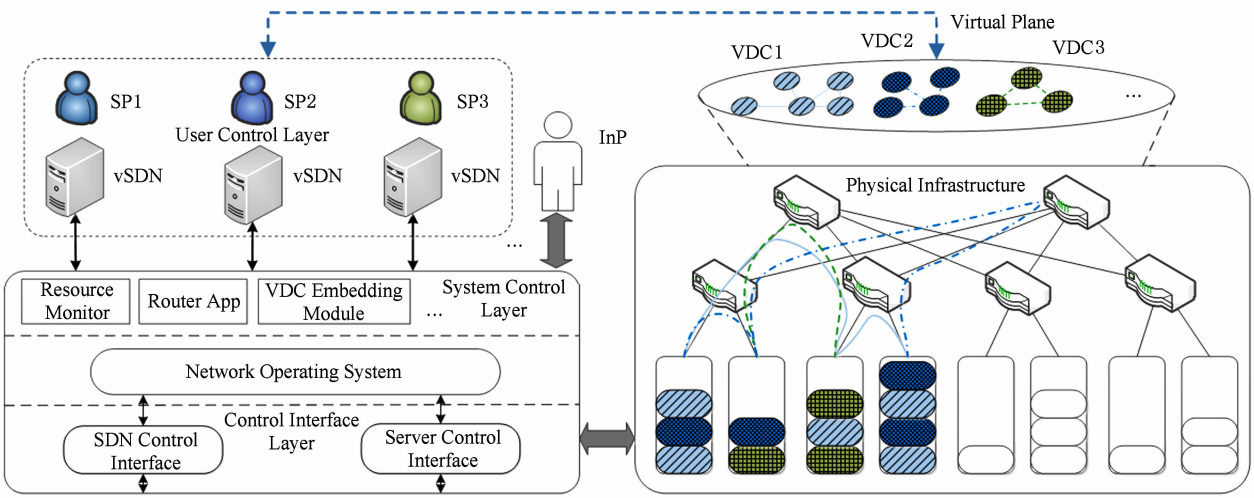


Fig. 1 The framework of resource management in SDN-enabled datacenter.

图 1 基于 SDN 的数据中心资源管理框架

3) 网络操作系统(NOS). ①NOS 通过控制层的北向接口,对底层物理资源进行统一的管理和调度,可以利用 SDN 控制器下发流表策略,利用 Server 控制器操作物理主机和虚拟机;②NOS 内部会维护底层网络的全局拓扑视图以及各个节点和链路的分配、使用状态;③NOS 抽象化底层物理资源的使用方式,为上层应用程序(系统控制程序和用户控制程序)提供统一的接口,并且为不同的应用程序提供隔离并分配权限;④NOS 负责事件的分发:当 NOS 收到底层网络产生的事件时,它会根据事件的源、类型、优先级(系统级或用户级)以及订阅者信息,将事件分发给相应的应用程序.

4) 系统控制层. 系统控制层包含运行在 NOS 之上协助 InP 实现数据中心自动化管理的应用程序,比如路由程序、VDC 映射程序、资源监控程序等.

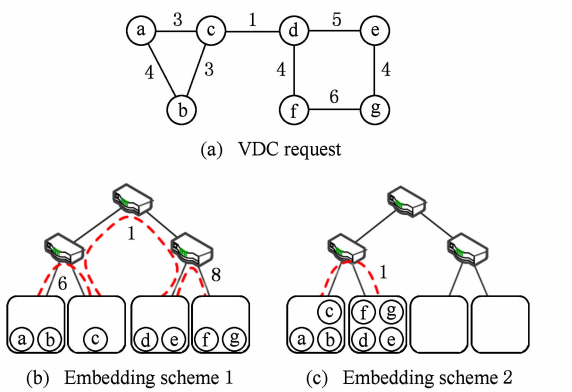
5) 用户控制层. 对 SP 来说,它只能看到自己所拥有的 VDC,并且可以像使用真实的 SDN 网络一样来控制它,底层物理网络对 SP 来说是透明的. SP 可以通过自己的控制器 vSDN 对自己的 VDC 进行完全控制. 举例来说,SP 在 VDC 上部署了一个 Web

服务,其中有面向 Web 用户的即时流量,也有数据库节点之间的备份流量. 为了能给用户提供更优质的服务,SP 可以为不同流量定义不同的优先级:面向用户的高优先级流量、后端数据库的低优先级备份流量. SP 可以通过 vSDN 控制器将优先级转发策略发送给 NOS,NOS 将该策略转化为合法流表项下发到底层交换机中. SP 仅有操作自己拥有的 VDC 的权限,没有操作其他 VDC 和物理资源的权限.

1.2 资源利用率与可靠性

VDC 映射问题不仅可以帮助 InP 实现数据中心的自动化管理和资源的灵活分配,而且可以达到多方面的目标,比如提高资源利用率、提高 VDC 请求的接受率、最大化收益. 但是目前为止,考虑 VDC 的可靠性问题,折衷可靠性需求与映射代价进行资源映射问题却没有被很好地解决. VDC 的可靠性在云计算环境中是一个非常重要的问题. 对 SP 来说,服务的中断,虽然只有几秒,可能会带来巨大的损失,严重影响 SP 的声誉[7]. 同时,由于服务不可达,InP 也需要向 SP 赔偿一定的罚金,比如亚马逊 EC2 承诺 1 个月内服务的在线率高于 99%,但低于 99.95%会赔偿所付总额的 10%[8].

在数据中心中,一台主机可以放置多台虚拟机. 如果 2 台虚拟机被分配到同一台物理主机中,那么就无需为它们之间的虚拟链路分配资源;如果 2 台虚拟机被分配到了不同的物理主机中,那么就需要为它们之间的虚拟链路分配物理链路资源. 图 2(a)表示一个 VDC 请求,其中ⓐ~ⓖ代表虚拟节点,虚拟节点之间的实线表示虚拟链路,实线上的数字表示虚拟链路的带宽需求。图 2(b)(c)表示 2 种映射方案,其中数字表示虚拟链路所占用的带宽. 为了提高映射后 VDC 的可靠性,VDC 中的虚拟机应该分散到多个故障域,如图 2(b)所示. 但这种情况下,



(a) VDC request

(b) Embedding scheme 1 (c) Embedding scheme 2

Fig. 2 Different VDC embedding schemes.

图 2 不同映射策略的对比

VDC 会占用更多的链路资源,资源利用率较低. 另一方面,如果为了提高资源利用率,VDC 中的虚拟机应该尽量放置在一台物理主机或者一台机架内. 图 2(c)的 VDC 映射方案占用较少的链路资源,具有较高的资源利用率,但如果一台物理主机出现故障,就有可能导致大量 VM 失效. 在实际中,数据中心中存在多种类型的故障,比如物理主机故障、ToR (top of rack)交换机故障、电力故障等. 本文只考虑物理主机故障.

目前已经有大量针对 VN 映射问题的研究. 文献[9]采用了一种 top-k 支配模型,通过平衡多种资源因素对节点排序. NodeRank[10] 同时考虑节点资源容量、链路带宽容量以及 VN 的拓扑对节点排序,实现拓扑感知的节点映射. 然而,只有少量针对虚拟数据中心映射问题的研究. Zhani 等人[11] 提出了 VDC planner 的映射框架,可以通过虚拟节点的迁移提高 VDC 的接受率,但它并没有考虑 VDC 的可靠性问题. SecondNet[12] 采用一种贪心算法映射 VDC,但它限制了一台物理主机最多只能映射一个 VM. Greenhead[2] 研究了分布在不同地理位置的多数据中心中的 VDC 映射问题,它根据 VDC 拓扑以及 VM 的位置限制条件将 VDC 划分为多个组,使用贪心算法将各个组分配到不同数据中心中.

2 问题建模

2.1 网络建模

本文中,我们将底层数据中心网络通过无向图 $G(N \cup X, E)$表示,其中 N 表示数据中心中的主机集合,X 表示数据中心中的交换机集合,E 表示物理节点(主机和交换机)之间的链路集合. 每台主机都包含一系列的资源属性(比如 CPU、内存等),本文使用 $A=\{a_i, a_2, \cdots, a_k\}$表示,其中 a_i 表示主机的第 i 种属性. $c_i(n,t)$表示在时刻 t 主机 $n(n \in N)$中资源 $a_i(0<i \leqslant k)$的剩余容量. 每条链路有一定的带宽容量,$bw(e,t)$表示链路 e 在时刻 t 的剩余带宽容量.

对 VDC 请求 j,本文使用无向图 $G^j(N^j, E^j)$表示. 我们用 t_j 表示它的到达时间,用 T_j 表示它在数据中心内的生命时长,用 r_j 表示 VDC 的可靠性需求. 在只考虑主机故障的情况下,VDC 的可靠性定义为在最坏情况下的 VM 的存活率或可达率. VDC_j 映射后的实际可靠性应该不小于 r_j,具体为

$$r_j \leqslant \frac{|N^j| - \max\limits_{n \in N} P_j(n)}{|N^j|}, \tag{1}$$

其中,$P_j(n)$表示 VDC_j 分配到主机 n 中的 VM 数量.根据式(1),对于 VDC_j,一台主机上允许放置的最大 VM 数目 K 可以计算为

$$K=\left\lfloor |N^j|[1-r_j]\right\rfloor. \tag{2}$$

2.2 VDC 映射过程

为了方便问题描述,本文定义了 3 个二元决策变量 x_j,$y_i(n',n)$和 $z_j(e',l)$.x_j 表示 VDC_j 是否被接受:

$$x_j=\begin{cases}1, \text{ if } VDC_j \text{ 被接受};\\ 0, \text{ otherwise.}\end{cases} \tag{3}$$

$y_i(n',n)$表示 VDC_j 中的 VM $n'(n'\in N^j)$是否被分配到物理主机 $n(n\in N)$中:

$$y_j(n',n)=\begin{cases}1, \text{ if } n' \text{ 被分配到主机 } n;\\ 0, \text{ otherwise.}\end{cases} \tag{4}$$

$z_j(e',l)$表示 VDC_j 中的虚拟链路 $e'(e'\in E^j)$是否被分配到物理路径 l 中($l\in P$,P 表示数据中心内的所有无向无环路径):

$$z_j(e',l)=\begin{cases}1, \text{ if } e' \text{ 被分配到路径 } l \text{ 上};\\ 0, \text{ otherwise.}\end{cases} \tag{5}$$

在给出映射过程需要满足的限制条件之前,为了方便描述,本文使用 $\pi_j(t)$表示 VDC_j 在时刻 t 是否有效:

$$\pi_j(t)=\begin{cases}1, \text{ if } x^j=1 \text{ 且 } t_j<t\leqslant t_j+T_j;\\ 0, \text{ otherwise.}\end{cases} \tag{6}$$

1) 如果 VDC_j 被拒绝,即 $x^j=0$,那么对任意 VM 和虚拟链路来说,$y_i(n',n)$和 $z_j(e',l)$只能为 0:

$$\begin{aligned}&y_j(n',n)\leqslant x_j,\forall n'\in N^j,n\in N,\\ &z_j(e',l)\leqslant x_j,\forall e'\in E^j,l\in P.\end{aligned} \tag{7}$$

2) 主机资源和链路带宽的容量限制条件:

$$\sum_j\sum_{n'\in N^j}\pi_j(t)y_j(n',n)c_i(n')\leqslant c_i(n,t),\quad \forall n\in N, \tag{8}$$

$$\sum_j\sum_{l\in P}\sum_{e'\in E^j}\pi_j(t)z_j(e',l)\delta_{e,l}bw(e')\leqslant bw(e,t),\forall e\in E. \tag{9}$$

其中 $\delta_{e,l}$表示物理链路 e 是否为物理路径 l 的一部分.

3) 保证被接受 VDC 的所有元素都被映射:

$$\sum_{n\in N}y_j(n',n)=x_j,\ \forall j,n'\in N^j, \tag{10}$$

$$\sum_{l\in P}z_j(e',l)=x_j,\ \forall j,e'\in E^j. \tag{11}$$

4) 满足可靠性需求:

$$\sum_{n'\in N^j}y_j(n',n)\leqslant\left\lfloor |N^j|(1-r_j)\right\rfloor,\quad \forall j,n\in N; \tag{12}$$

5) 端点条件.如果虚拟链路 e'被分配到物理路径 l 中,那么 e'的 2 个端点应该分别分配到 l 的 2 个端点上.如果用 $u(\cdot)$和 $v(\cdot)$表示一条链路或路径的 2 个端点,那么需要满足条件:

$$\begin{aligned}z_j(e',l)\leqslant\ & y_j(u(e'),u(l))y_j(v(e'),v(l))+\\ & y_j(u(e'),v(l))y_j(v(e'),u(l)).\end{aligned} \tag{13}$$

2.3 收益模型

对每个被接受的 VDC 请求,SP 根据服务需求按照使用时间长短付费,而单位时间的价格由 VDC 所请求的资源量决定,定义如下:

$$R_j=\frac{\alpha}{k}\sum_{n'\in N^j}\sum_{i=1}^{k}\hat{c}_i(n')+(1-\alpha)\sum_{e'\in E^j}\widehat{bw}(e'), \tag{14}$$

其中,$\hat{c}_i(n')$表示 VM n'对第 i 种资源的归一化需求量,$\widehat{bw}(e')$表示虚拟链路 e'的带宽归一化需求量,$\alpha(0<\alpha<1)$表示节点资源和链路资源的平衡因子.那么,从时刻 0~t,InP 可获得的收益为

$$R(t)=\int_0^t\sum_j\pi_j(t)R_j\mathrm{d}t=\sum_jR_j\times\int_0^t\pi_j(t)\mathrm{d}t. \tag{15}$$

将式(6)代入式(15)中:

$$\begin{aligned}R(t)=&\sum_{j\in\{j|t>t_j\}}R_j\int_{t_j}^{t}x_j\mathrm{d}t=\sum_{j\in\{j|t>t_j\}}R_j\int_{t_j}^{t_j+T_j}x_j\mathrm{d}t-\\ &\sum_{j\in\{j|t_j<t<t_j+T_j\}}R_j\int_{t}^{t_j+T_j}x_j\mathrm{d}t=\sum_{j\in\{j|t>t_j\}}R_jx_jT_j-\\ &\sum_{j\in\{j|\pi_j(t)=1\}}R_jx_j(t_j+T_j-t).\end{aligned} \tag{16}$$

$R(t)$可以表示时刻 0~t 已接受的 VDC 在它们生命时长到达后所产生的总收益减去当前存活的 VDC 还可以再产生的收益.

2.4 目标函数

本文的目标是帮助 InP 提高长期的单位时间平均收益,其定义如下:

$$\text{maximize } R=\lim_{t\to\infty}\frac{R(t)}{t}=\lim_{t\to\infty}\frac{\sum_{j\in\{j|t>t_j\}}R_jx_jT_j}{t}. \tag{17}$$

可以看出这个优化函数是一个非线性整数优化问题,该 NP-hard 问题的求解空间庞大,因此本文将提出一种启发式映射算法,最大化 InP 收益.

影响长期单位时间平均收益的因素主要有 2 个:VDC 接受率和收益代价比.VDC 接受率是指一段时间内 InP 接受的 VDC 请求数目占收到 VDC 请求的总数目的比例.一般来说,VDC 接受率越高,InP 就能获得更高的收益.收益代价比是指 VDC 请求的资源量与 VDC 实际占用资源量的比值,反映了物理资源的利用率.低收益代价比的 VDC 表示 VDC 占用了更多的物理链路资源,使物理网络产生

更多的碎片资源.这会进一步影响后续 VDC 的收益代价比,降低 InP 的接受率.

3 基于拓扑势的 VDC 映射算法

为了满足 VDC 需求,以最小代价为 VDC 分配资源,提高算法的 VDC 接受率和收益,本文提出一种启发式算法解决 VDC 的映射问题.算法的理论基础是利用拓扑势来评价网络中节点的重要性,在考虑 VDC 可靠性需求的基础上对虚拟机节点聚类,尽量将节点间流量较多的 VM 映射到相同的主机上,节点间流量较少的 VM 分散到不同的主机上.由于算法利用拓扑势实现节点映射(nodes mapping based on topological potential),因此我们将本文的算法命名为 NMP.本节首先介绍拓扑势的含义以及计算方法,然后详细描述本文的算法设计.

3.1 拓扑势

在物理学中,"势场"可以描述没有直接接触的节点之间所产生的相互作用,比如电磁场和重力场.拓扑势[13]就是受此启发,将类似的方法应用到网络中,评价网络节点的拓扑重要性.在网络中,我们定义 2 个节点之间的相互作用与它们的资源容量和它们之间的距离有关.本文使用高斯函数的形式描述 2 个节点的相互作用.一个节点的拓扑势等于网络中所有节点对它作用的叠加,节点 n 的拓扑势通过以下方式计算:

$$\varphi(n)=\sum_{n'\in N}\frac{1}{|A|}\sum_{i=1}^{k}\hat{c}_i(n')\times \mathrm{e}^{-\left(\frac{d_{nn'}}{\sigma\cdot\widehat{bw}_{nn'}}\right)^2}, \tag{18}$$

其中,$\hat{c}_i(n')$表示节点 n'中第 i 种资源的归一化容量,对物理主机来说表示剩余资源容量,对虚拟机来说表示请求的资源容量;e 为自然常数;$d_{nn'}$和$\widehat{bw}_{nn'}$表示节点 n 和 n'之间最短路径的距离和归一化带宽容量;σ 是影响因子,表示一个节点的影响范围大小.在网络中,拓扑势最大的节点可以说明以这个节点为中心的范围内拥有更多的资源.

选择一个合适的 σ 对拓扑势来说十分重要.如果 σ 太大,每个节点的影响范围就很大,那么网络中所有节点的拓扑势就会非常接近,难以区别节点的重要性;相反,如果 σ 太小,一个节点的拓扑势只由自身的资源容量决定,邻居节点的影响近乎消失.根据信息熵的理论,如果所有节点的拓扑势几乎一样,那么所提供的信息量就最小,拓扑势的熵最大.因此我们可以通过计算拓扑势的最小熵获取合适的影响因子 σ.拓扑势的熵计算为

$$H=-\sum_{n\in N}\frac{\varphi(n)}{\Phi}\ln\left(\frac{\varphi(n)}{\Phi}\right), \tag{19}$$

其中,$\Phi=\sum_{n\in N}\varphi(n)$.

3.2 算法设计

在本节中,我们会详细描述 NMP 算法的具体细节.NMP 算法的主要思想是采用一种迭代的方式,每次搜索一簇 VM 节点,使簇内 VM 相互之间具有更密集的链路带宽,在满足 VDC 可靠性需求的条件上,将这簇 VM 集合映射到一台主机中.这可以有效地减少虚拟链路占用的资源.算法的具体步骤如下:

步骤 1. 根据 VDC 的可靠性需求,计算单台物理主机所允许的最大 VM 数量 K.

步骤 2. 通过虚拟机节点聚类的方式搜索一簇 VM 节点.首先选择拓扑势最大的 VM 节点作为簇心.由式(18)对拓扑势的定义可知,网络节点拓扑势的大小由该节点与其邻居节点共同决定,反映了一个节点在网络中的重要性.拓扑势最大的节点可以作为簇心实现网络节点聚类,所以在每次迭代过程中使用未分配的 VM 集合中拓扑势最大的节点作为簇心搜索 VM 簇.

步骤 3. 从未分配的 VM 集合中选择可以使网络模块度最大化的 VM,加入到 VM 簇中,重复执行直到 VM 簇的大小到达 K.模块度可以评价网络社区划分结果的好坏.模块度将社区内链路密度与社区间链路密度做比较.一个优秀的社区划分结果在社区内相比于社区之间具有更紧密的链路,所以本文利用模块度来选择 VM 加入到 VM 簇,以使簇内具有更高的链路密度.有关模块度的计算以及虚拟机节点聚类的方法将在 3.2.2 节详述.

步骤 4. 为了映射当前搜索到的 VM 簇,需要选择一台物理主机.为了减少 VDC 请求所占用的链路资源,本文提出了 2 种主机选择方法:1)主机的选择限定在一簇物理主机集合中,而不是数据中心内的所有主机.本文通过网络节点聚类,搜索一簇相邻且剩余资源容量较高的物理主机集合.2)根据主机的剩余资源容量与到已映射 VM 的距离对所有主机评分,选择评分最高的主机.具体的主机选择算法在 3.2.2 节描述.

步骤 5. 如果选择的主机不满足当前 VM 簇的需求,则需要减小当前 VM 簇的大小.本文以添加到 VM 簇的相反顺序删除 VM,直到 VM 簇可以被映射到主机中.

步骤 6. 如果仍有未映射的 VM,回到步骤 2 重复执行.

下面的算法 1 给出了 NMP 算法的具体步骤.

算法 1. NMP 算法的具体步骤.

输入:$G(N\cup X,E)$,$G^j(N^j\cup X^j,E^j)$;

输出:是否映射成功.

符号说明:U 表示未映射的 VM 集合,C 表示当前 VM 簇集合.

① $U\leftarrow N^j$;
② $C\leftarrow\varnothing$;
③ 根据可靠性需求计算 K;
④ 计算 U 中所有 VM 的拓扑势;
⑤ while $U\neq\varnothing$ do
⑥ $v\leftarrow U$ 中拓扑势最大的 VM;
⑦ $C\leftarrow\{v\}$;
⑧ $stack\leftarrow[v]$;
⑨ for $i\leftarrow 0$ to $K-1$ do
⑩ 计算 U/C 中所有 VM 对应的模块度增量 ΔQ;
⑪ $v'\leftarrow\Delta Q$ 最大的 VM;
⑫ $C\leftarrow C\cup\{v'\}$;
⑬ $stack.push(v')$;
⑭ end for
⑮ $U\leftarrow U-v'$;
⑯ $s\leftarrow$选择一台最佳主机;
⑰ while s 的节点资源和出口带宽不满足 C 且 $stack\neq\varnothing$ do
⑱ $last\leftarrow stack.pop()$;
⑲ $C\leftarrow C-last$;
⑳ end while
㉑ $U\leftarrow U\cup\{last\}$;
㉒ if $C==\varnothing$ then
㉓ return false;
㉔ end if
㉕ 将 C 映射到 s 中;
㉖ end while
㉗ return true.

NMP 算法包含 2 个最关键的步骤:虚拟机节点聚类和物理主机的选择.接下来本文对这 2 部分进行详尽的描述.

3.2.1 虚拟机节点聚类

在网络中,拓扑势最大的节点可以说明以这个节点为中心的范围内拥有更多的资源,所以本文将拓扑势最大的节点作为搜索 VM 簇的核心.模块度用来评价网络社区划分算法的结果.模块度的定义为[14]

$$Q=\frac{1}{2m}\sum_{v,w}\left[A_{v,w}-\frac{k_vk_w}{2m}\right]\delta(c_v,c_w),\quad(20)$$

其中,$A_{v,w}$ 表示节点 v,w 之间的链路权重;$k_v=\sum_w A_{v,w}$ 表示节点 v 的度;c_v 表示节点 v 所属的社区;$m=\frac{1}{2}\sum_v k_v$ 表示所有节点度之和的一半.如果定义 $e_{ij}=\frac{1}{2m}\sum_{v,w}A_{v,w}\delta(c_v,i)\delta(c_w,j)$ 表示社区 i 与 j 之间的链路所占比例,定义 $a_i=\frac{1}{2m}\sum_v k_v\delta(c_v,i)$ 表示至少有一个端点在社区 i 中链路的比例,那么模块度可以写成另一种方式:

$$Q=\sum_i(e_{ii}-a_i^2),\quad(21)$$

这表示模块度等于社区内链路所占比例减去随机情况下社区内链路所占比例.

在算法 1 中,每次迭代过程利用拓扑势最大的 VM 节点作为核心搜索一簇 VM 集合.最开始,C 中只有核心 VM 节点;接下来,从 U/C 中选择一个 VM 加入到 C 中并可以最大化网络整体模块度,重复添加 VM 直到 C 的大小达到 K.为了提高算法的效率,本文只计算将一台 VM 从集合 U 中移到 C 中的模块度增量 ΔQ.如果将一个节点 v 从类簇 j 中移到类簇 i 中,ΔQ 可以计算为

$$\Delta Q=e_{ii}-a_i^2+e_{jj}-a_j^2-\left[e_{ii}+\frac{k_v(i)}{2m}-(a_i+\frac{k_v}{2m})^2\right]-\left[e_{jj}-\frac{k_v(j)}{2m}-(a_j-\frac{k_v}{2m})^2\right]=\frac{1}{2m}\left[k_v(i)-k_v(j)+2k_v(a_j-a_i)-\frac{k_v^2}{m}\right],\quad(22)$$

其中,$k_v(i)=\sum_w A_{v,w}\delta(c_w,i)$ 表示其中一个端点为 VM v,另一个端点属于类簇 i 的链路数量.如果每次迭代过程保存着集合 U 和 C 对应的 e_{ii} 和 a_i 值并不断更新,那么 ΔQ 可以更高效地计算出来.

3.2.2 物理主机选择方法

搜索到一簇 VM 节点后,我们需要选择一台物理主机来映射 VM 簇.为了减少簇间虚拟链路的跳数,选择物理主机时应该同时考虑主机的剩余资源容量以及到已映射 VM 的距离.本文提出了 2 种选择物理的方法:1)通过节点聚类(clustering)的方法将 VDC 的映射限定在一簇资源容量较多的主机集合中,本文将这种方法命名为 NMPC;2)根据主机的拓扑势和到已映射 VM 的距离(distance)对主机进行评分,并选择评分最高的主机,这种方法被命名为 NMPD.下面详细说明这 2 种方法:

1) NMPC.为了方便描述,我们用 CS 表示正在搜索的一簇候选主机集合.初始时刻 $CS=\varnothing$.首先,

计算所有物理主机的拓扑势;接下来,将拓扑势最大的主机移入到 CS 中;然后通过迭代的方式,每次从剩余主机中选择与 CS 中主机最近且拓扑势最大的主机,将其移入到 CS 中,直到 CS 集合的大小和资源容量足够映射 VDC. 当每次搜索到一簇 VM 集合后,选择 CS 中拓扑势最大的主机来映射这簇 VM 集合.

2) NMPD. 与 NMPC 先搜索一簇候选主机集合,再从其中选择主机的方式不同,NMPD 根据主机的拓扑势和到已映射 VM 的距离对所有未考虑的主机进行评分,并选择评分最高的主机. 本文通过式(23)计算每台物理主机的评分.

$$score(n) = tp_n \times \frac{1}{\sqrt[\lambda]{\frac{1}{|us|}\sum_{n' \in us} d_{nn'}}}, \tag{23}$$

其中,tp_n 表示物理主机 n 的拓扑势,us 表示已映射的 VM 所在的物理主机集合,$d_{nn'}$ 表示 2 台物理主机的距离,λ 是主机拓扑势和到已选择主机平均距离的平衡因子. 被评价的主机拓扑势越大,到已选择主机的平均距离越近,评分越高.

4 实验结果与分析

4.1 仿真环境

我们基于 Python 开发了一个离散事件仿真器来模拟数据中心中的 VDC 映射过程. 数据中心网络采用 6 端口的 FatTree 拓扑,包含 45 个交换机、54 台物理主机. 每台主机包含 16 个 vCPU 和 8 096 MB 内存,交换机的每个端口的带宽容量为 1 000 Mbps. VDC 请求使用 BRITE[15] 拓扑生成器生成. VDC 请求的 VM 数量均匀分布在 10～50 之间,每台 VM 请求的 vCPU 均匀分布在 1～4 之间,内存容量均匀分布在 512～2 048 MB,每条虚拟链路请求的带宽容量均匀分布在 100～200 Mbps. VDC 请求的到达服从均值为 0.03 的泊松分布,VDC 的生命时长服从均值为 500 单位时间的指数分布. 本文选择基于节点排序的 NodeRank[10] 作为对比算法,并扩展到 VDC 场景,允许一台物理主机可以映射 VDC 的多个 VM,并满足可靠性需求,该对比算法在后文中均以 Baseline 标记.

4.2 评价指标

本文的主要目标是提高 InP 的长期单位时间平均收益. 影响单位时间平均收益的因素主要有 2 个:接受率和收益代价比. 除了接受率、收益代价比外,本文还提出了核心链路利用率的指标来比较各种算法的性能. 下面对各种评价指标进行详细说明.

1) 接受率. 时刻 t 的接受率定义为在时刻 t 被接受的 VDC 数目占收到的 VDC 请求总数的比例:

$$Accept(t) = \frac{\sum_{j \in \{j | t > t_j\}} x_j}{|\{j \mid t > t_j\}|}, \tag{24}$$

其中,$|\cdot|$ 表示集合中的元素数目. 只有接受率并不能完全反映算法的性能. 如果一种算法只接受小规模请求、拒绝大请求,可能会产生较高的接受率,收益反而会减小. 这是因为接受率是由接受的 VDC 数目决定的,而收益是由 VDC 规模大小和接受的 VDC 数目共同决定的.

2) VDC 收益代价比. 对 VDC_j 来说,收益代价比表示 VDC 的请求资源量与实际占用资源量的比值:

$$RC_j = \frac{R_j}{C_j}, \tag{25}$$

其中,代价 C_j 由其实际所占用的资源量决定:

$$C_j = \frac{\alpha}{k}\sum_{n' \in N^j}\sum_{i=1}^{k}\hat{c}_i(n') + (1-\alpha)\sum_{e' \in E^j} hops(e')\,\widehat{bw}(e'). \tag{26}$$

3) 累积收益代价比. 对数据中心来说,数据中心在时刻 t 的收益代价比为数据中心在时刻 $0 \sim t$ 获得的累积收益与产生的代价的比值,反映了数据中心的资源利用率. 具体定义如下:

$$RC(t) = \frac{R(t)}{C(t)} = \frac{\int_0^t \sum_j \pi_j(t) R_j \mathrm{d}t}{\int_0^t \sum_j \pi_j(t) C_j \mathrm{d}t}. \tag{27}$$

低的收益代价比一般由 2 方面的因素导致:① 不合理的映射算法所导致的资源浪费;② 数据中心内剩余资源有限或资源碎片较多. 低收益代价比的 VDC 会占用更多的链路带宽资源,进一步造成更多的资源碎片,这会影响将来 VDC 的资源分配与收益代价比,会降低将来时刻的接受率,最终影响 InP 的长期收益.

4) 核心链路占用率. 核心链路是指由核心层交换机所连接的链路. 核心链路不仅承载着数据中心内日益加剧的东西向流量,也负责通往外部的南北向流量. 实际研究[16] 发现核心层交换机的链路负载相当高,这会导致数据中心网络拥塞甚至丢包. 式(28)定义了核心链路平均占用率来评价 VDC 映射算法的表现.

$$CU(t) = \frac{1}{|E_c|}\sum_{l \in E_c}\frac{1 - bw(l,t)}{BW(l)}, \tag{28}$$

其中,$E_c \in E$ 表示核心链路集合,$BW(l)$ 表示物理链路 l 的总带宽容量.

4.3 NMPD 中平衡因子的选择

在 3.2 节中,NMPD 算法同时考虑主机的拓扑势和到已选择主机的距离,对每台主机评分并选择评分最高的主机. 其中,λ 表示拓扑势与距离因素之间的平衡因子. λ 越大,距离因素影响越小,会导致虚拟链路占用更多的带宽资源;相反,λ 越小,距离因素的权重越大,距离已映射 VM 稍微远一点的主机就会被忽略掉. 图 3 反映了接受率随平衡因子 λ 的变化关系. 从图 3 可以看到,λ 太大或太小,接受率都比较小,当 $\lambda=1.5$ 时,映射算法的接受率最高. 因此接受率随 λ 的变化并不是很大:当 λ 从 1 增加到 4,接受率变化范围只有 0.03. 本文在 NMPD 算法中默认使用 $\lambda=1.5$.

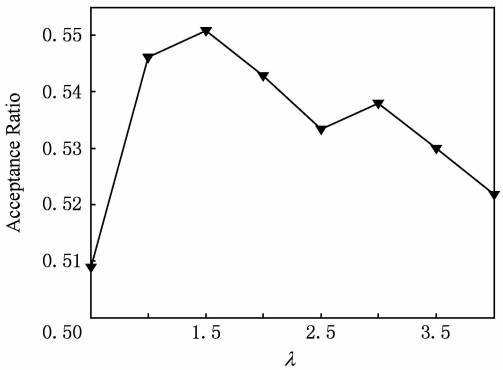


Fig. 3 Acceptance ratio with different λ.

图 3 接受率随 λ 的变化

4.4 不同算法的比较

首先,本文将 VDC 的可靠性需求设为 0.2~0.9,对比了 3 种算法在不同评价指标下的表现.

1) NMPC 接受最多的 VDC 请求. 从图 4 可以看到,NMPC 和 NMPD 算法具有较高的 VDC 接受率和收益,而且 NMPC 的表现优于 NMPD. 这表明 NMPC 算法可以接受最多的 VDC 请求,也获得最高的收益. 举例来说,在 20 000 单位时间时,NMPC 和 NMPD 的 VDC 接受率分别为 0.58 和 0.55,比 Baseline 算法的 0.46 高 10%以上.

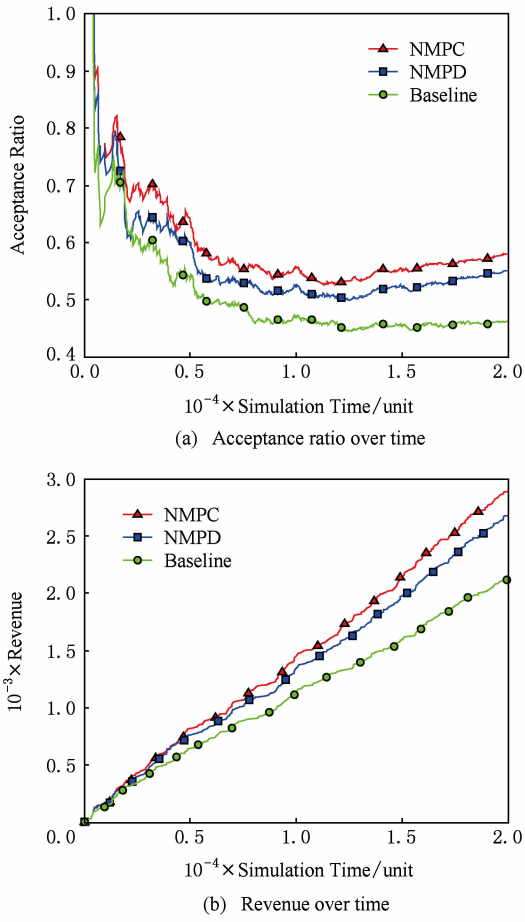


Fig. 4 Acceptance ratio and revenue over time.

图 4 接受率与收益随时间的变化

2) NMPD 具有最好的资源利用率. 尽管 NMPC 算法可以接受最多的 VDC 请求,然而它的收益代价

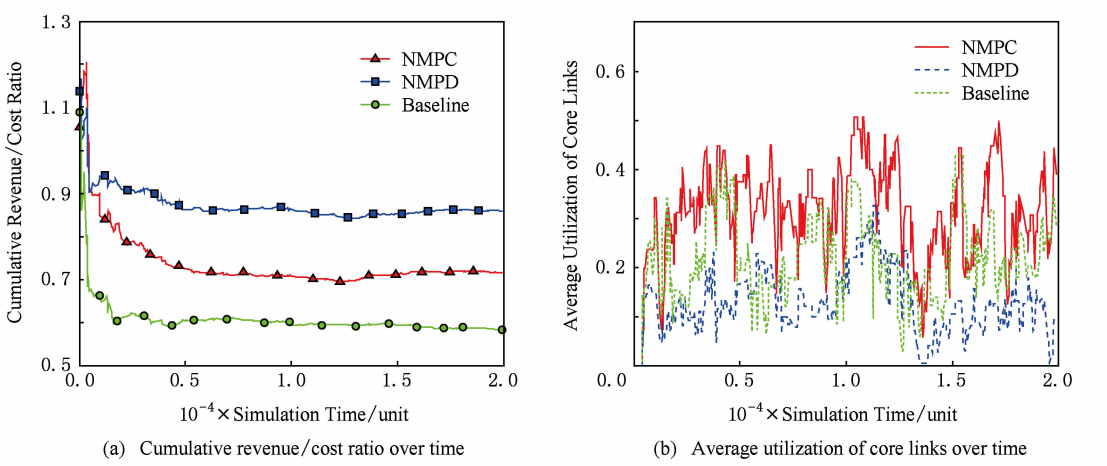


Fig. 5 Cumulative revenue/cost ratio over time and average utilization of core links over time.

图 5 累积收益代价比与核心链路平均占用率随时间的变化

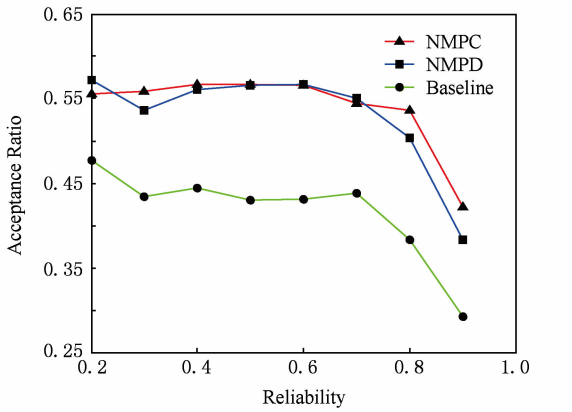
比却低于 NMPD. 如图 5(a)所示,大部分时间 NMPD 的累计收益代价比接近 90%,大幅高于 NMPC 的 70%和 Baseline 算法的 60%. 这表明 NMPD 以最小的代价为 VDC 分配资源,可以提高数据中心的资源利用率. 当数据中心剩余资源有限或者碎片资源较多时,VDC 的映射代价太高,NMPD 会拒绝掉新到的 VDC 请求;相反,NMPC 和 Baseline 算法由于贪婪的本质,尽力映射每一个请求,这样导致收益代价比较低.

3) NMPD 核心链路占用率最小. 图 5(b)显示大部分时间 NMPD 算法的核心链路平均占用率要低于另外 2 种算法;大部分时间内,NMPD 算法的核心链路平均占用率要比 NMPC 低 20%左右;NMPC 算法的核心链路平均占用率甚至高于 Baseline,这是因为 NMPC 算法比 Baseline 接受了更多的请求. NMPD 算法比 Baseline 接受了更多的 VDC 请求,但却占用了更少的核心链路带宽,这表明了 NMPD 算法可以达到较高的资源利用率.

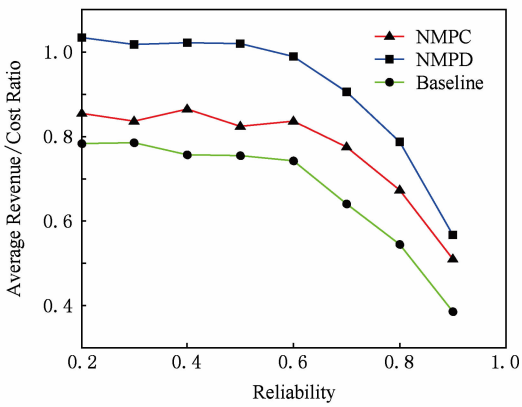
4.5 可靠性需求与请求到达率对算法的影响

这节本文分别对比了不同 VDC 可靠性需求和不同请求到达率对算法的影响.

1) 不同 VDC 可靠性需求对算法的影响. 从图 6 可以看到,算法的接受率和 VDC 平均收益代价比都会随着 VDC 可靠性需求的增加而下降. 这是因为可靠性需求越大,算法需要将 VDC 分散到更多的物理主机中,导致 VDC 会占用更多的链路资源. 图 6(a)展示了接受率与可靠性需求的变化关系,可以看到可靠性需求在 0.2～0.7 之间时,NMPC 和 NMPD 的接受率几乎一致,均比 Baseline 高 10%左右;当可靠性需求超过 0.7 后,NMPD 算法的接受



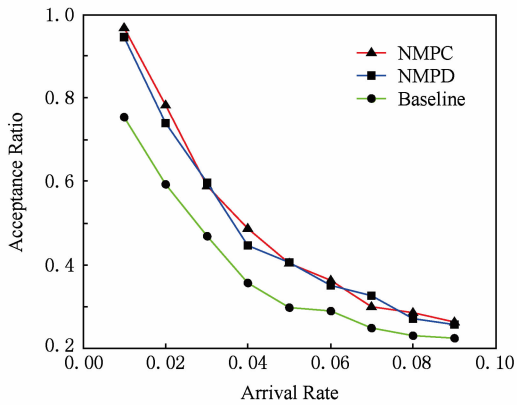
(a) Acceptance ratio with reliability



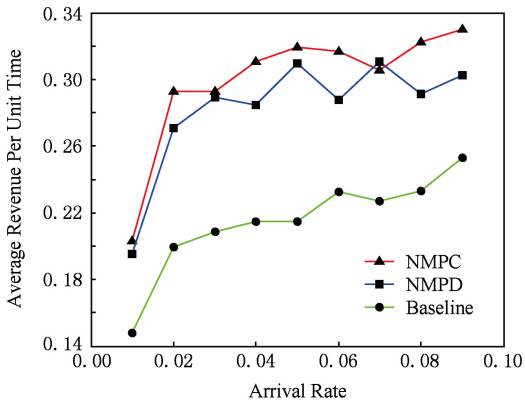
(b) Average revenue/cost ratio with reliability

Fig. 6 The influence of reliability requirement.

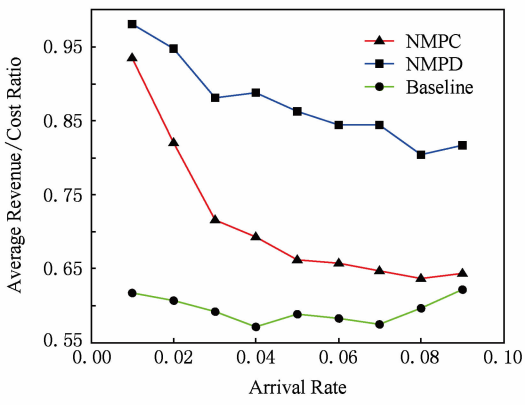
图 6 可靠性需求对算法的影响



(a) Acceptance ratio with arrival rate



(b) Average revenue with arrival rate



(c) Average revenue/cost ratio with arrival rate

Fig. 7 The influence of VDC arrival rate.

图 7 请求到达率对算法的影响

率迅速下降,而 NMPC 在超过 0.8 后才剧烈下降.这是因为 NMPD 更倾向于拒绝占用资源较多的 VDC 请求.从图 6(b)可以再次印证上面的分析,NMPD 算法维持着较高的 VDC 平均收益代价比,而且 VDC 平均收益代价比随着可靠性需求的增加而下降.

2) 不同 VDC 请求到达率对算法的影响.由图 7(a)(b)可见,随着请求到达率的增加,各种算法的接受率会不断下降,然而单位时间的平均收益具有上升趋势且会趋于平衡.很显然,随着请求到达率不断增加,虽然更多的请求被拒绝,但底层网络接受的请求数也在增加,所以单位时间的平均收益也会增加.由于底层网络资源的固有容量限制,单位时间的平均收益不可能无限增长下去,所以会趋于平衡.从图 7(c)可以发现,VDC 平均收益代价比会随着到达率的增加而下降.这是因为算法接受了更多数目的 VDC 请求,此时底层网络剩余资源有限,只能以更大的代价接受新到的 VDC 请求.

4.6 利用收益代价比门限进一步优化算法

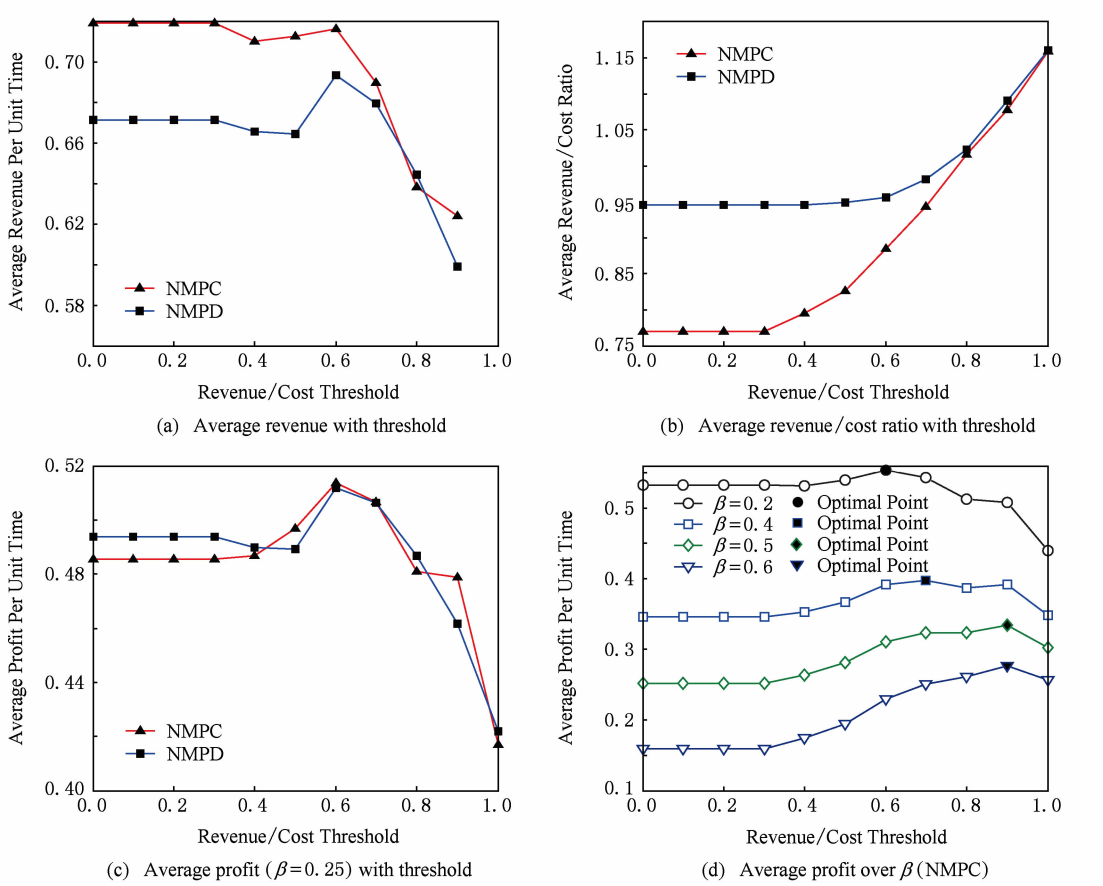
虽然 NMPD 算法通过一种启发式的方法拒绝了一些低收益代价比的 VDC 请求,但本文所提到的算法都具有贪婪的本质:只要可以发现满足 VDC 请求需求的资源,它就会接受 VDC 请求.如果 VDC 请求占用了大量的底层资源,这将会影响后续 VDC 的收益代价比和接受率,形成恶性循环.基于以上分析,本文尝试基于收益代价比门限值,在算法中采用一种动态监控策略,通过比较门限阈值大小动态接受或拒绝 VDC 请求:如果 VDC 请求的收益代价比低于门限值,则拒绝掉该 VDC 请求.

为了更方便地评价收益代价比门限对算法的影响,本文定义了一个新指标——单位时间内的平均利润,具体如下:

$$profit(t)=\frac{R(t)-\beta\times C(t)}{t}, \tag{29}$$

其中,β 表示收益和代价的调节因子.本文定义收益 $R(t)$ 与代价 $C(t)$ 时,资源请求量和占用量均做了归一化处理.在实际中,这么计算得到的收益和代价存在着一定的比例,所以使用 β 作为调节因子.

针对本文所提出的 2 种算法,我们对比了不同收益代价比门限情况下 2 种算法的表现,如图 8 所示.图 8(a)显示了单位时间的平均收益与收益代价比门限的变化情况.从图 8(a)可以看到,NMPD 的



(a) Average revenue with threshold

(b) Average revenue/cost ratio with threshold

(c) Average profit (β=0.25) with threshold

(d) Average profit over β (NMPC)

Fig. 8 Performance with different revenue/cost thresholds.

图 8 不同收益代价比门限对算法的影响

单位时间平均收益随门限值的增加先增大后减小.因为合适的门限值可以拒绝掉一些浪费较多资源的VDC请求,接受更多高收益代价比的VDC请求.但如果门限值太高的话,被拒绝掉的请求也会越多,导致接受率下降,单位时间平均收益减少.对NMPC来说,门限值为0～0.6,其单位时间平均收益并没有提高,然而NMPC所映射VDC的平均收益代价比却大幅提升,如图8(b)所示.这表明当门限值从0变到0.6时,NMPC以更少的代价获得了相同的收益.

图8(c)(d)显示收益代价比门限与单位时间平均利润的关系.从图8(c)可以看到,2种算法的单位时间平均利润(β=0.25)很接近,均随门限值的增加先增大后减小;当收益代价比门限值为0.6时,二者均达到最优的单位时间平均利润.图8(d)显示了NMPC算法中不同β、门限值与单位时间平均利润的关系.从图8(d)可以看到,单位时间的平均利润均随门限值的增加先增大后减小.当β较大时,表示VDC占用的物理资源具有较高的成本,所以最优的收益代价比门限也越高;当β较小时,表示物理资源的成本较低,较低的收益代价比门限就可以获得最优的单位时间平均利润.在实际中,可以根据资源成本与收益的比例设置合适的门限值,以获得最优的利润.

5 结束语

本文提出了一种基于SDN的数据中心资源管理框架;同时以提高InP收益为目标,折衷VDC可靠性需求与映射代价,提出了基于拓扑势的VDC映射算法.为了减少链路资源的占用,本文通过虚拟机节点聚类的方式,将链路密集的虚拟机集合分配到同一台主机中,将链路稀疏的虚拟机节点分散到不同主机中.本文提出了2种不同的主机选择方法,进一步提高了物理资源的利用率.通过仿真实验,我们发现本文所提的算法可以达到较高的接受率与收益.最后,我们提出一种动态监控策略,选择收益代价比高的VDC请求,通过选择合适的收益代价比门限,可以进一步优化算法.在将来的研究中,我们会考虑分布式数据中心中的VDC映射问题.

参 考 文 献

[1] Bari M F, Boutaba R, Esteves R, et al. Data center network virtualization: A survey [J]. IEEE Communications Surveys & Tutorials, 2013, 15(2): 909-928

[2] Amokrane A, Zhani M F, Langar R, et al. Greenhead: Virtual data center embedding across distributed infrastructures [J]. IEEE Trans on Cloud Computing, 2013, 1(1): 36-49

[3] Kim H, Feamster N. Improving network management with software defined networking [J]. IEEE Communications Magazine, 2013, 51(2): 114-119

[4] Zhang Chaokun, Cui Yong, Tang Heyi, et al. State-of-the-Art survey on software-defined networking [J]. Journal of Software, 2015, 26(1): 62-81 (in Chinese)
(张朝昆, 崔勇, 唐翯祎, 等. 软件定义网络(SDN)研究进展[J]. 软件学报, 2015, 26(1): 62-81)

[5] Dong shi, Li Ruixuan, Li Xiaolin. Energy efficient routing algorithm based on software defined data center network [J]. Journal of Computer Research and Development, 2015, 52(4): 806-812 (in Chinese)
(董仕, 李瑞轩, 李晓林. 基于软件定义数据中心网络的节能路由算法[J]. 计算机研究与发展, 2015, 52(4): 806-812)

[6] Fischer A, Botero J F, Till Beck M, et al. Virtual network embedding: A survey [J]. IEEE Communications Surveys & Tutorials, 2013, 15(4): 1888-1906

[7] Rabbani M G, Zhani M F, Boutaba R. On achieving high survivability in virtualized data centers [J]. IEICE Trans on Communications, 2014, 97(1): 10-18

[8] Amazon. Amazon EC2 service level agreement [EB/OL]. 2013 [2015-12-17]. http://aws.amazon.com/cn/ec2/sla

[9] Li Xiaoling, Wang Huaimin, Ding Bo, et al. Resource allocation with multi-factor node ranking in data center networks [J]. Future Generation Computer Systems, 2014, 32: 1-12

[10] Cheng Xiang, Su Sen, Zhang Zhongbao, et al. Virtual network embedding through topology-aware node ranking [J]. ACM SIGCOMM Computer Communication Review, 2011, 41(2): 38-47

[11] Zhani M F, Zhang Q, Simon G, et al. VDC planner: Dynamic migration-aware virtual data center embedding for clouds [C] //Proc of IFIP/IEEE Int Symp on Integrated Network Management. Piscataway, NJ: IEEE, 2013: 18-25

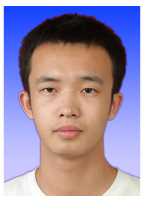
[12] Guo C, Lu G, Wang H J, et al. SecondNet: A data center network virtualization architecture with bandwidth guarantees [C] //Proc of the 6th Int Conf. New York: ACM, 2010: 15:1-15:12

[13] Li D, Du Y. Artificial Intelligence with Uncertainty [M]. Boca Raton, FL: CRC, 2007

[14] Newman M E J, Girvan M. Finding and evaluating community structure in networks [J]. Physical Review E, 2004, 69(2): 026113

[15] Medina A, Lakhina A, Matta I, et al. BRITE: An approach to universal topology generation [C] //Proc of the 9th Int Symp on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. Piscataway, NJ: IEEE, 2001: 346-353

[16] Benson T, Anand A, Akella A, et al. Understanding data center traffic characteristics [J]. ACM SIGCOMM Computer Communication Review, 2010, 40(1): 92-99



Wen Xuemin, born in 1988. PhD candidate at the Institute of Information Engineering, Chinese Academy of Sciences. His main research interests include data center network, network virtualization and cloud computing.

Han Yanni, born in 1981. PhD and assistant researcher in the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences. Her current research interests include resource management, network virtualization and future Internet.

Yu Bing, born in 1990. PhD candidate at the Institute of Information Engineering, Chinese Academy of Sciences. Her main research interests include service migration and resource management (yubing@iie.ac.cn).

Sun Jianpeng, born in 1989. Master candidate at the School of Information Science and Engineering, Shandong University. His main research interests include virtual machine live migration in cloud computing (sunjianpeng@iie.ac.cn).

Xu Zhen, born in 1976. PhD, professor, PhD supervisor. Member of China Computer Federation. His main research interests include database security, trusted computing and network security (xuzhen@iie.ac.cn).