清 华 大 学

综 合 论 文 训 练

题目：数据中心云服务中多租户网络带宽保障研究

系　　别：计算机科学与技术系

专　　业：计算机科学与技术

姓　　名：王靖易

指导教师：尹霞教授

2015年 6月10日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____ 导师签名：_____ 日　期：_____

中文摘要

基础设施即服务（IaaS）是现在云数据中心所提供的一项重要服务，通过虚拟化技术，不仅节省租户对日常维护基础设施的开销，而且获得了更好的可扩展性和灵活性。而云供应商也能通过复用来最大化利用数据中心的计算和网络资源从而降低成本。这样的优势使得云数据中心的IaaS服务越来越被重视，而同时对这一服务也就有了更高的需求。租户网络是IaaS供应商给租户提供的网络资源，现在的租户网络普遍缺乏对网络性能的保障，仅仅是尽力而为的网络，而租户对网络性能却是有着要求。为了保证租户网络的性能，于是需要研究租户网络的带宽保障这一关键技术。

当前有不少关于租户带宽保障的研究，但现有的模型均没有解决用户需求动态变化带来的种种问题，大多数研究或避开这一问题或只是简单处理，而用户的带宽保障需求在实际应用中确实会产生变化。所以，需要一种新的带宽保障解决方案既能方便描述用户的需求，同时解决现在其他方案不能解决的用户需求变化的问题。

本文章主要进行了如下工作；

1. 提出了一个新的租户带宽保障模型，能够便于描述租户复杂的带宽保障需求，同时本身将带宽资源和计算资源解耦的特点使得实现这一模型的租户带宽保障系统能够有效的处理租户需求的动态变化。设计实现了应用此模型进行租户虚拟机部署的算法。
2. 设计实现了一个基于云的测试流量生成系统，该系统具备高可扩展、高并发、准确、易于使用的特点。在实际的OpenStack云平台上部署了该系统，并验证了系统的性能和可扩展性。该系统作为一个典型的IaaS租户应用，可以获益于上面提出的租户带宽保障方案，同时验证租户带宽保障方案的能力。
3. 对设计实现的算法进行了功能和性能测试，结果显示本文提出的模型可以使租户灵活地权衡取舍几种资源，从而达到比一般模型更好的性能。

关键词：云；数据中心网络；多租户，带宽保障

ABSTRACT

Infrastructure as a Service(IaaS) is an important type of service of cloud data centers. Taking advantage of the virtualization technology, tenants can save the cost of daily server maintaining and get clusters with more scalability and flexibility. Cloud providers can maximize the utility of computation and network resources of data centers through multiplexing of DCN. These benefits make contributes to the increasing popularity of IaaS. However, the more we use, the more requirements arise. The best effort tenant networks lack the guarantee of performance which tenants need. To protect the performance of tenant network, more research about the tenant bandwidth guarantee problem is needed.

Prior works and models can't handle the issue raised by tenant requirement extending. However, the demand of tenant will change commonly in real tenant application scenarios. A new approach is needed to resolve the issue of tenant requirement extending.

The main work of this thesis include:

1. Propose a new model of tenant bandwidth guarantee which called Elastic Bandwidth Guarantee model to resolve the problem of tenant requirement extending by decouple the computation and network resources of tenant demand. Design a algorithm of VM allocation of this model.
2. Implement a cloud-based test traffic generation system which is scalable, high-performance, accurate and easy-to-use.Deploy the system on OpenStack cloud and evaluate the performance of the system.
3. Conduct simulations on the algorithm of EBG model. The result shows that the EBG model is more flexible to use the resource of cloud.

Keywords: cloud; DCN; multi-tenancy; bandwidth guarantee

目 录

主要符号对照表

VM	虚拟机 (Virtual Machine)
DC	数据中心 (Data Center)
BW	带宽 (Bandwidth)
cloud	云数据中心
IaaS	基础设施即服务 (Infrastructure as a Service)

第 1 章　引言

1.1　研究背景

基础设施即服务（Infrastructure as a Service，IaaS）的数据中心网络，或者称作——云，是如今一种重要的网络形态。云服务供应商提供存储、计算、网络基础设施，通过对基础设施的虚拟化合理复用，减少成本提供廉价的基础设施。随着基础设施即服务（IaaS）的云数据中心的发展，越来越多企业、个人用户将自己的服务器放置在云端，省去了自行维护机房的开销，虚拟化技术也帮助这些租户获得灵活性、扩展性更好的服务器集群。

租户是IaaS云上的用户，通过支付费用获得一定的计算、存储、网络资源。对于租户来说，云提供的应当是一个与真实网络具备一样功能的基础设施，同时云供应商还能够让租户方便自由的定义自己的网络，如同使用实际物理设备一样。虽然同一个云上可能同时有大量的租户存在，但每个租户能够得到的都是云所提供的一个租户网络的抽象，对租户来说如同只有自身在使用一个独立的网络。而实际上租户却是共用一个云上的资源的，这就产生了租户网络带宽保障问题。

随着应用的深入，现有的IaaS服务提供给租户的“尽力而为”网络已经不能满足需要。越来越多的研究发现[1–5]，在虚拟化的租户网络上，网络性能十分不稳定，且不同虚拟网络之间互相干扰，造成租户网络的性能十分差，拥塞频繁发生，延迟高且波动大。租户带宽保障问题就由此提出，租户通过支付额外费用，获得受保障的带宽，云供应商通过带宽保障方案，确保租户的受保障带宽来满足租户对网络性能隔离的要求。

租户网络由于虚拟化和复用，在数据中心网络上形成了复杂的网络使用情境，本身IaaS云为节省成本就会尽可能最大化利用网络资源，而虚拟化和多租户又加剧了网络资源的耗尽，使得保障网络性能比起传统网络面临更多问题。为了保障租户网络的性能，需要尽可能的针对云数据中心这样特殊虚拟化环境做出相应的优化方案，来提高网络性能，才能使租户的网络性能得到保障。

租户的带宽保障问题已有很多相关研究，而实现一个租户带宽保障方案需要很多方面工作。已有的研究中提出的方案虽然能够部分解决租户带宽保障的

问题，但仍有改进余地，并且对于租户需求变化这一情况，少有研究加以考量。而在实际应用中，租户租用的虚拟机集群，很有可能随着需求增加而扩容，租户会租用更多数量的虚拟机，请求更大的带宽。在之前的研究中，对于这种租户网络扩容的情况很少提及或只是简单处理，并没有深入进行讨论。而租户需求变化不仅常见，也会带来一系列新问题，是租户带宽保障问题中不得不面对的。为了解决现有方案没有考虑的问题，需要提出一种新的解决方案，这个方案不仅要继承原有方案的优点，同时还要处理租户需求变化带来的一系列问题。

同时，为了深入理解租户带宽保障问题，并对所提出的新方案加以验证，需要一个实际的应用场景。本文设计实现了一个基于云的测试流量生成系统，这个系统有着实际的用途，并且部署在云的虚拟机上，是一个典型的租户网络应用。这个系统本身对网络性能的高要求，使得其十分适合用来验证租户带宽保障解决方案。

1.2 主要工作

本文对租户带宽保障问题进行了研究，主要进行了以下工作：

1. 对租户带宽保障问题进行了综述，综述了租户网络性能、租户带宽保障的模型、租户带宽保障系统的相关研究；
2. 设计了新的租户带宽保障模型EBG，通过新的模型，克服了之前方案的缺点，能够更好的表达租户的需求便于租户使用，同时新模型自身的性质使得新模型更灵活，节省网络资源，并能解决租户需求动态变化带来的问题，实现了模型的部署算法；
3. 设计实现了基于云的测试流量生成系统，实现了一个高可扩展、并发性能高、易于使用的基于OpenStack云的测试流量生成系统。该系统能够生成极高负载的流量，同时可以方便的通过虚拟化技术添加虚拟机的方式对系统进行扩展；
4. 进行了对新租户带宽保障方案的性能评价模拟实验，并结合实际的应用场景进行了模拟实验，验证了新模型的性能。

1.3 论文结构

本文共包含六章：

- 引言，介绍研究背景和主要工作；
- 相关研究综述，对租户带宽保障相关研究进行了综述；
- 弹性带宽保障模型及部署算法，提出了租户需求变化产生的问题，提出了新的租户带宽保障模型，设计实现了模型的部署算法；
- 基于云的测试流量生成系统，针对几个设计目标设计实现了一个真实的系统，对系统进行了性能测试和评价；
- 模拟实验与性能评价，针对提出的模型和算法，进行了模拟实验进行性能评价，同时结合设计实现的租户应用，对带宽保障模型进行了评价；
- 总结。

第 2 章　相关研究综述

2.1　引言

本章对租户带宽保障问题的相关研究进行了综述，首先介绍多租户下云数据中心的性能问题，然后介绍租户带宽保障的相关研究，最后重点介绍现有的租户带宽保障模型。

2.2　多租户下的数据中心网络性能

IaaS云服务通过虚拟化技术，提供租户以计算和网络资源，通过高度复用，来提高效率增加收益。而正是由于多租户并行复用和虚拟化，使得云数据中心出现了很多一般网络没有的性能问题。

数据中心网络由于其特殊性，有着高拥塞的特点[6,7]，大量并发的流量争抢网络带宽， Incast现象[8]造成严重拥塞，导致网络性能严重下降。而IaaS云上，由于虚拟化，还加重了网络的问题。

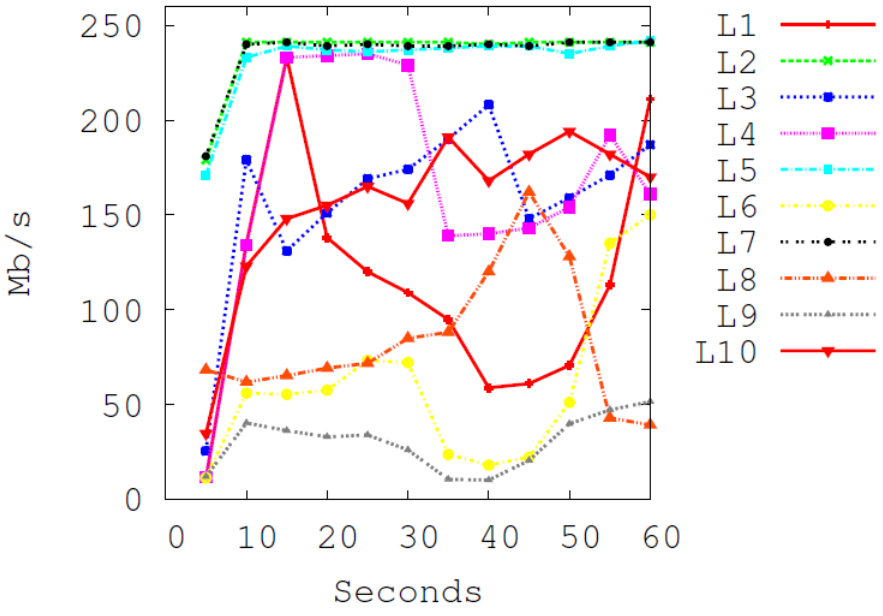


图 2.1　云数据中心中不同流量带宽差异大且波动大

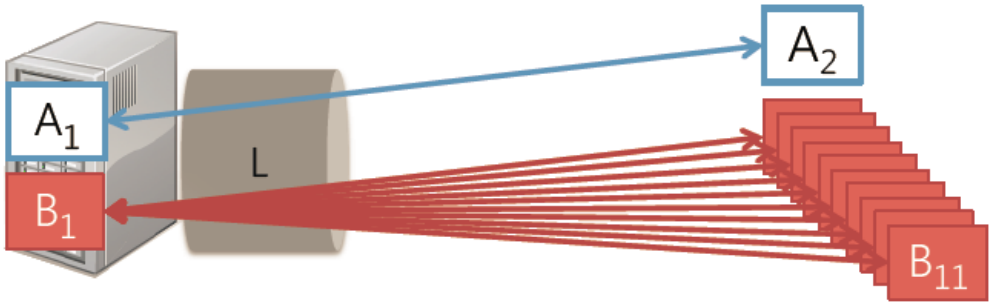


图 2.2　一个租户会抢占其它租户的网络资源

在[1,5,9]等研究中提出了，虚拟化会造成云数据中心中，网络的性能十分不稳定。 IaaS云数据中心中吞吐率波动极大，延迟有着长尾特性，一些网络流的延迟比其它流高出数倍[5]。在这些以实际数据中心为对象进行的测量研究中，均发现带宽的不稳定以及延迟的波动性，如图2.1中所示，不同的流的带宽差异巨大，并且同一条流的带宽波动也极大。这些研究表明，虚拟化在一定程度上造成了这个问题，由于虚拟机管理层（hypervisor）的调度，会造成共用相同物理资源的虚拟机性能上互相干扰，一台虚拟机的网络流量会由于hypervisor调度导致暂时被停止，从而造成延迟。

而除了虚拟化带来的问题，高度复用也导致了多租户网络的问题。由于IaaS数据中心上大量租户的同时存在[10]，租户间公享网络资源必然导致互相争抢网络资源，而传统的网络体系中只有“流公平性（flow fairness）”而没有“租户公平性”的概念，就会造成租户间带宽分配的不平衡，如图2.2所示，一个发起多条流的租户会完全抢占其它租户的网络资源[2]。甚至一个恶意的租户可以利用这个问题影响其它的租户，比如使用高速率的UDP流，建立大量TCP流[10]。

如今的虚拟化技术使得CPU、存储等资源都可以充分隔离并按支付费用比例分配，而网络资源缺乏这种隔离手段，导致多租户的IaaS云数据中心面临着网络性能的问题。而租户又是需要网络性能保障来保障其高层次需求如任务完成时间、用户访问延迟等。所以租户网络的性能保障问题是如今一个十分重要的课题，需要进行深入的研究。

2.3 租户带宽保障相关研究

在多租户的环境下，租户虽然在逻辑上各自享有独立的网络，但实际上是共用同一个物理网络，而网络上的瓶颈链路出现拥塞，租户的实际网络通信带宽就会出现极大的波动，有的租户的带宽会被其他网络通信完全抢占，使得租户的网络完全瘫痪。对于网络延迟有很高要求的租户，提供带宽的最低保障是十分必要的。为租户提供最低带宽保障，就能让租户即使在最差的情况下，也能拥有一定的网络性能保障，租户就能够估计其通信的最大延迟，从而满足租户高层次的需求。

对于租户带宽保障这一问题，有很多相关研究，而为了提供带宽保障，设计很多不同方面的工作。总体来说，包含以下方面：首先是带宽保障的模型，通俗的说就是以什么样的形式提供给租户保障，或者说租户和供应商间如何表达带宽保障需求；其次是准入控制（admission control）及虚拟机分配（VM allocation），租户请求虚拟机及带宽保障，提交给供应商后，要判断当前数据中心能否有足够资源提供给租户，并计算将虚拟机放在什么物理位置上（或者说将何处的VM提供给租户），既能满足带宽保障的条件，又能尽可能节省网络带宽资源；第三是带宽保障实现，即通过一定手段保障租户的带宽，目前主要有带宽预留和速率分配两种，这类研究首先要实现租户的带宽保障，然后在这一基础上，要尽可能的提高性能，充分利用数据中心的资源。

大部分相关研究涵盖了本节开头所述的多个方面，形成了一个比较完整的租户带宽保障方案，下面介绍比较前沿且重要的几个研究。

2.3.1 预留带宽的保障方案

为了保障租户的带宽，最简单直接的方法就是将带宽静态的分配给租户独享。租户带宽保障最初研究提出的方案基本都属于这一类[3,11]，然而这种方法虽然简单，但是问题也十分明显，就是对网络资源的浪费十分严重。所以有了之后的工作保留的带宽保障方案。

2.3.2 工作保留（work-conserving）的保障方案

将链路带宽的一部分直接预留给租户，虽然能方便简单的保障租户的带宽，但并不是最有效的方案。由于数据中心网络的特性，租户并不会一直使用其带宽，流量往往是突发的[7,12]，这样直接预留的方案就会造成大量的带宽浪费。

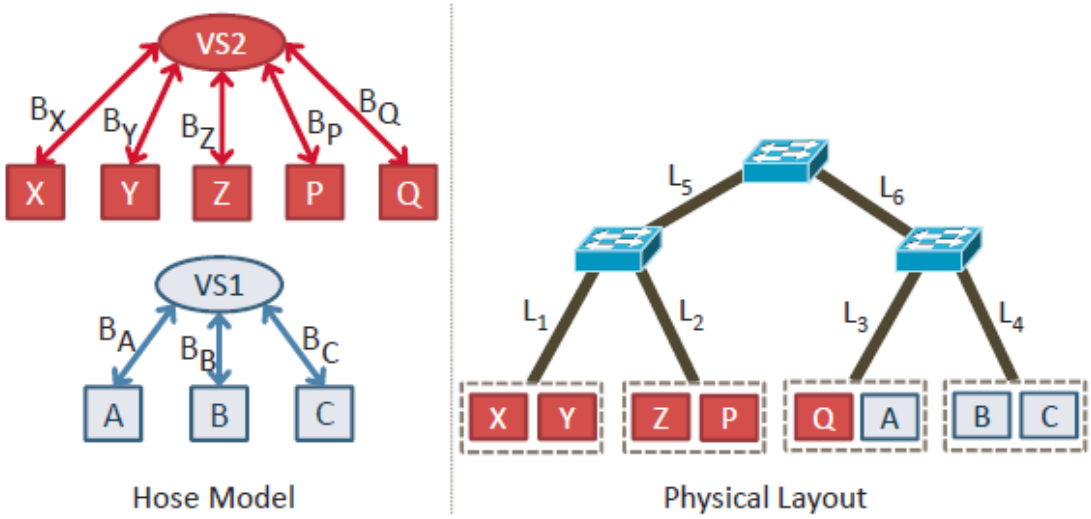


图 2.3　一个带宽保障需求的示意图

如果一个租户在不使用其带宽时，这部分带宽能够给其它租户使用，就能大幅提高网络的使用率，节省带宽降低成本。工作保留（work-conserving）概念由此提出，工作保留指网络的带宽使用是尽可能用尽的，租户分配的带宽在不使用时能够给其他租户使用，网络中的瓶颈链路带宽被完全占满[2,13]。一个能够合理充分利用网络带宽资源的租户带宽保障方案应当是工作保留的。

很多新的方案由此提出[13–16]，这些方案充分利用了数据中心网络的特性，在租户不使用其带宽时，这部分带宽能够给其他租户使用，从而将数据中心的网络资源充分使用，不会造成浪费，节省了成本。下面介绍几个典型研究。

ElasticSwitch[14]这一研究提出了一种工作保留的租户带宽保障解决方案。ElasticSwitch采用的是hose model，其输入是每个租户VM的带宽最低保障以及VM的物理分布和网络拓扑、链路带宽信息，这个研究并不考虑VM分配和准入控制的问题，只考虑如何速率控制。

如图2.3。红色和蓝色代表两个不同租户，在hose model下，每个租户对于每台VM得到B_{VM}的带宽保障，而实际上在物理网络中，这些VM可能共用主机和链路，甚至不同租户共用主机和链路。这样，就需要将hose model提供的带宽保障转化为物理网络上的带宽分配策略。

ElasticSwitch的研究将这一问题分为两部分，保障分割（guarantee partitioning，GP）和速率分配（rate allocation，RA）。GP的功能是将用户配置的带宽保障分割为每一个发送接收VM对之间的带宽保障，将简单的hose model变为端到

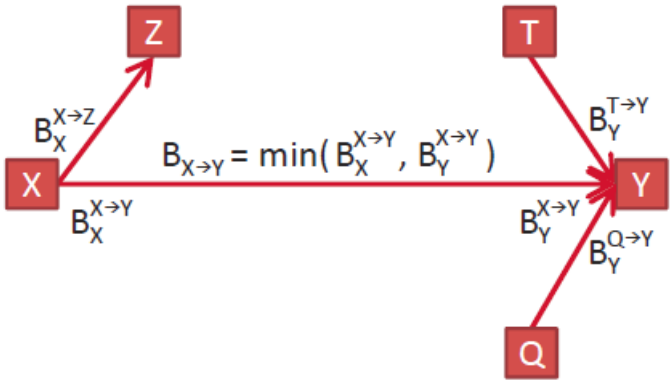


图 2.4　GP的例子

端的pipe model。端到端的带宽保障设置为源端发送带宽保障与目的端接收带宽保障的最小值，同时同一个源到多个不同目的的带宽可按照不同原则分配（一般是平均分配），在某一对源目的之间的带宽受到接收端带宽限制时，可以相应增加同一个源到其他目的的带宽。这样的分割带宽方式，保证了整个网络带宽保障不超出实际网络限制，同时尽可能节省了带宽（默认前提是输入的hose model带宽保障是可以被网络接受的，这属于准入控制问题）。GP实际实现是通过间隔一定时间动态调整的方式分割带宽，目的端通过专门的控制报文告知发送端其分配到的带宽，发送端根据取最小值的原则设定带宽，每次通过接收端检测实际使用的带宽，如果一个源目的对完全使用了分配的带宽且没有达到可分配的上限（一般就是同一个源按照公平原则分配的带宽），就增大带宽保障，如果没有用到，则减少带宽保障。图2.4是一个GP的例子。

图中，假设虚拟机X, Y所设定的带宽保障都是B，X到Z和X到Y的带宽保障本来平分都是$B/2$，而$B_X^{X\to Y}$受$B_Y^{X\to Y}$的限制实际为$B/3$，通过动态调整，$B_X^{X\to Z}$就可以从$B/2$增加到$2B/3$。

RA部分的功能是实际的控制速率，RA通过类似于weighted TCP[17]的算法，实现速率的控制和分配，最终达到在保障带宽的前提下，尽可能的利用剩余带宽，剩余带宽按照带宽保障的比例来分配（即weighted TCP中的权值就设为带宽保障的值）。而由于传统的weighted TCP并没有考虑带宽保障的需求，所以需要进行一些修改，最重要的一点就是，在严格保障带宽和最大化利用网络之间存在权衡，若要充分利用网络剩余带宽，就会在突发流量到来时无法及时调整带宽分配导致带宽保障不能实现，为了解决这个问题，提出了三个修改，首先

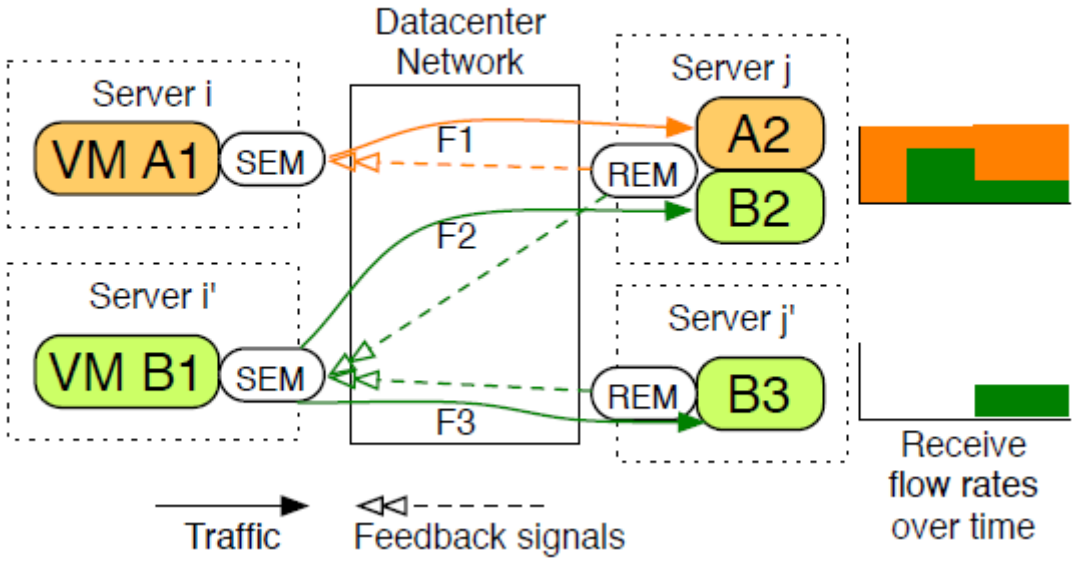


图 2.5　EyeQ的结构示意图

是设置余量（Headroom），令最大可提供的带宽保障小于链路最大容量（在试验中采用的是预留10%链路带宽）；第二是延迟增长（Hold-Increase），在出现拥塞时，将对应的VM到VM对的带宽增长暂停一段时间，这个时间反比于分配的保障；第三是速率警戒（Rate-Caution），当一个带宽分配增加到超过GP分配的带宽保障时，增长速率会降低。这些机制共同保证了，当提供给租户的带宽保障基本用尽链路带宽时，最后RA会将带宽分配收敛到每个VM到VM对的带宽基本等于其带宽保障。

EyeQ[15]这一研究也提出了一种带宽保障的解决方案。与ElasticSwitch的主要不同点在于，这一研究应用于高速数据中心网络（10G链路带宽），同时数据中心网络的拓扑在核心层交换机提供了足够冗余使得拥塞基本不发生在核心层交换机。 EyeQ分为Receiver EyeQ Module, REM和Sender EyeQ Module, SEM 两部分，分别在发送端和接收端的物理主机上（文中提到可以实现在网卡上也可以实现在hypervisor或者物理机系统内核上）。 SEM, REM的结构和工作方式如图2.5。

图右侧的带宽示意图表示的是两个主机Server j与Server j’的接收到的流的速率随时间的变化，颜色分别对应不同的流。起初只有F1一条从VM A1到A2的流，占满了Server j的全部带宽；之后从VM B1到B2的流F2开始，由于B1的带宽

保障，使得REM通过反馈报文告知Server i上的SEM进行限速，F2的速率达到了B1的带宽保障同时F1的速率下降；然后F3启动，由于同是属于VM B1的流，平分了B1的带宽保障，F2和F3平分了Server i’的带宽，而此时Server j的带宽就有了富余，于是在REM的反馈报文下，Server i再次通过SEM调整了速率限制，使得F1的速率上升，充分利用了Server j的带宽。

REM拥有测速功能，每隔一定间隔，统计出到达同一个VM的速率，SEM拥有限速功能，直接限制各个VM的发送速率。SEM处于发送端，所有发送出去的流，根据VM的带宽保障配置，可以直接通过SEM进行限速，而REM处于接收端，在发生竞争时，需要通过反馈报文，告知SEM，然后通过SEM的速率控制来实现带宽保障。SEM上的速率限制器，是一个分层结构，首先对于每个VM有一个根限制器，然后每个根下面有对于不同目的地址的叶子限制器，所以实际上EyeQ实现的也是基于VM到VM的pipe model带宽保障。SEM通过REM发送来的反馈信息（接收端的速率）动态调整发送端的速率限制，最终收敛到一个既能保障带宽又能充分利用链路性能的速率。

本质上ElasticSwitch与EyeQ实现带宽保障的方法基本一致，都是需要通过某种方式动态的调整VM的发送速率，通过速率限制，来实现带宽保障。而ElasticSwitch除了自己提出的在核心交换机会出现拥塞的情况下还能工作以外，主要的不同点在于强调了GP的功能。而如本章节开头所说，带宽保障这一问题实际上涉及了很多方面，这些研究均涉及了其中的一部分，而要实现完整的带宽保障系统，需要结合这些工作的优点。

2.3.3 其它类型的保障方案

在一些研究中[18,19]，没有使用上一节work-conserving的方案，但也考虑了租户带宽使用的变化。这类方案通过将租户的带宽使用建模为一个根据时间变化的函数，然后将这一时变函数放入数据中心中。这类研究虽然比起静态预留的方案有了进步，但是建模的方式比较复杂且不能准确表现租户的实际情况，而且也不能真正做到work-conserving。相比下，work-conserving动态调整速率限制的方案能够更好地贴合租户带宽的使用状况，更好地利用网络资源。

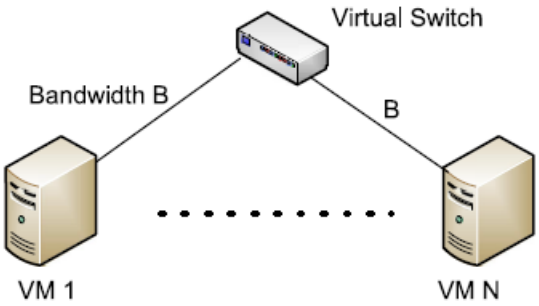


图 2.6　Hose model示意图

2.4　带宽保障模型介绍

租户带宽保障问题中，如何表达租户的需求，供应商如何提供租户带宽保障，需要通过带宽保障模型来体现。而模型又是和虚拟机分配与准入控制相关的，这一类研究是租户带宽保障中重要的一类研究。下面将介绍几种重要的租户带宽保障模型。

2.4.1　Pipe model

最直观的租户带宽保障模型就是pipe model，这个模型通过定义每个VM对之间的带宽，来定义租户的带宽需求。租户通过提供一个流量矩阵，来给出所有VM对之间的带宽使用需求。这一类研究中[11,20,21]，可以从租户的输入中直接确定每个VM的速率分配，之后的准入控制和虚拟机分配也比较容易。但缺陷也很明显，租户需要提供一个复杂的流量矩阵，不便于使用的同时，扩展性也差。并且，这一类虚拟机分配的方法只能通过暴力搜索的方式来找到合适的分配位置，虽然在不同研究中都有优化，但仍然复杂度较高。

2.4.2　Hose model

Hose model是当前最主流的模型，在[3]这一研究提出后，被后续的大量研究所使用，包括前面介绍的几种work-conserving的带宽保障方案。如图2.6，hose model将租户的网络抽象成一个简单的拓扑，所有虚拟机由一个虚拟交换机相连。而每个虚拟机连接的交换机的链路是被带宽保障的。更具体地说，每个虚拟机的带宽保障是B，则每个虚拟机能够发送B的流量，并且能够从其他虚拟机接收B的流量。

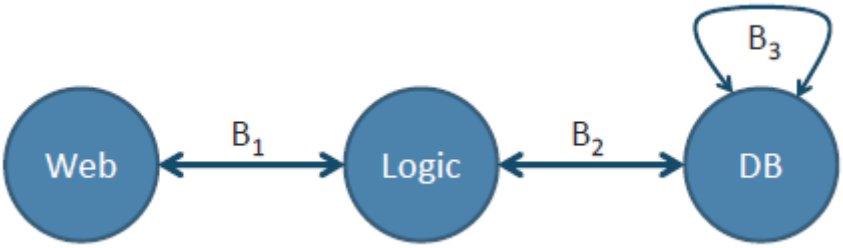


图 2.7　常见的互联网应用的通信结构

Hose model十分容易使用，因为租户只需要提供两个参数，虚拟机总数和每个虚拟机需求的带宽保障。而这个模型也十分容易部署，由于将租户的拓扑抽象为了这样的简单结构，所以在分配虚拟机时能够利用这个特点来实现启发式搜索，[3]研究中也利用模型的特点设计了简单的虚拟机分配算法。

由于其简单易用的优势，大量带宽保障方案使用这种模型[14,15]，但是也可以看到，这些方案中最终都将hose model转化成了pipe model，之后才能真正用来对虚拟机进行速率分配。所以hose model是对pipe model进行了抽象的一种更贴近租户使用情景的模型。

2.4.3　TAG model

Hose model有其简单的优势，但这也是其劣势。 TAG[22]这一研究就提出，对于复杂的租户应用场景，hose model简单的拓扑不能准确描述，从而不能更好的提供带宽保障。

这一研究提出了一种新的模型，租户应用图（Tenant application graph，TAG），该研究提出简单的hose model并不适应复杂的应用，而一般只适用于如Map-Reduce一类的简单all to all的通信模式。在云数据中心上，除了Map-Reduce类的应用外，一类普遍的应用是web应用，典型的结构如图2.7。

图中是一个web应用的典型通信结构，分为web（前端）, logic（逻辑），database（数据库）三部分功能组件，组件间有着互相通信，组件内也有通信，每一个组件可以由任意多的VM组成的。在这样的应用场景下，对于租户来说，很难用简单的hose model来描述带宽需求，尤其是组件间的通信。

而TAG模型就解决了这个问题，通过模型表达租户应用的通信结构。TAG是一个赋权有向图，其节点代表一个组件，组件可以由任意多VM组成的，TAG图的边代表从一个组件到另一个组件的通信的带宽保障。每条边包含两个

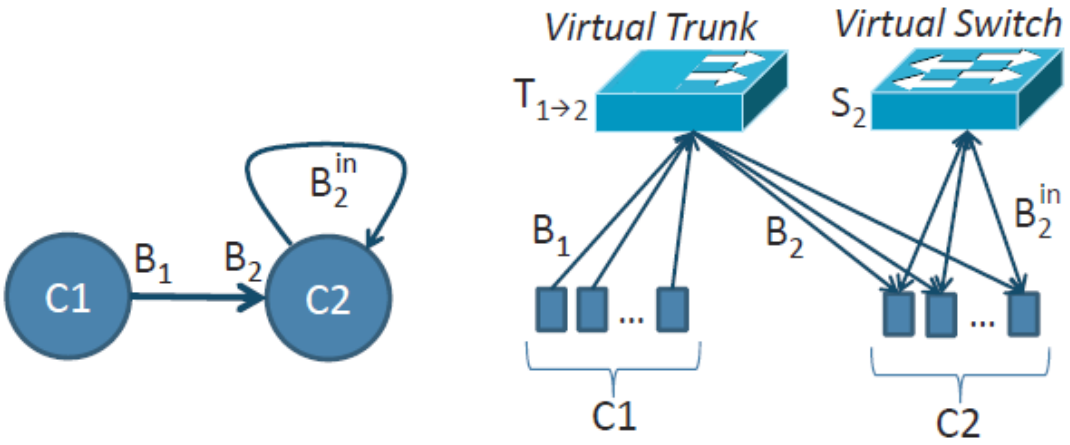


图 2.8　TAG模型的意义

参数$< S, R >$，有向边的方向表示网络通信的方向，S是发送方组件中每个虚拟机发送的带宽保障，R是接收方组件中每个虚拟机接收的带宽保障。而实际的两个组件间的通信带宽，由$min(S \cdot N_S, R \cdot N_R)$决定，$N_S$是发送方组件的VM数，$N_R$是接收方组件的VM数。TAG中还存在指向自身的边，表示组件内部通信的带宽保障，此时两个参数$S = R$，表示组件内每个VM的通信带宽保障。当一个TAG只包含有一个组件，且只有一条自指的边时，这个TAG等价于一个hose model。图2.8是一个TAG的例子。

图中C1, C2组件间有一条边，表示从C1的每一个VM到C2的每一个VM，都有B1的发送带宽和B2的接收带宽，C2指向自身的边表示C2内部所有VM间有B2的带宽保障。这个TAG可以转化为上图右边的类似于hose model的示意图，C1中的VM通过容量为B1的链路连接到交换机，再通过B2容量的链路连接到C2中的VM，同时C2中的VM处于一个B2容量链路的交换机下。

实际使用TAG时，需要有一个部署方法，即准入控制和虚拟机分配算法，这个研究提供了一个算法来实现部署TAG同时考虑通过将VM放置在合理的位置来节省带宽。这个算法的大致内容是，通过对树形物理网络自底向上的搜索，找到合适的位置放置VM，同时通过推导出的几个条件判断是否通过Colocation（将互相通信的虚拟机可能放在同一个物理机或子树）来节省带宽，得到一个解。

TAG模型的优点是能够更适合的表示应用的带宽需求，而可以看

到TAG是hose model的一个更高层抽象，它比hose model更加贴近租户的应用场景，所以也更容易使用同时能够更好的保障租户的带宽。

2.4.4 现有模型的缺陷

现有的租户带宽保障模型中，TAG是描述租户应用能力最强的，但是在所有模型及解决方案中，都没有考虑租户需求变化的情况，而租户需求变化却会引发一系列问题。本文将对这一问题进行深入研究，并给出解决方案。

2.5 小结

虚拟化环境下的多租户网络，是一个复杂的系统，同时又是一个如今益发重要的研究对象，随着云的普及，更多的企业用户会选择使用IaaS服务来替代自身管理服务器, 由于云数据中心的特性和租户网络间的互相影响，租户网络的带宽保障是十分重要的研究课题，如何实现租户带宽保障是需要深入研究的。

已有研究在租户带宽保障上有很多解决方案，然而仍有很多不足，有着改进的空间。本文将会对这一问题提出自己的见解，并给出更好的解决方案。

第 3 章　弹性带宽保障模型及部署算法

3.1　引言

已有研究已经提出了几种不同的租户带宽保障模型，但仍存在缺陷，本文首先提出由租户需求变化引发的问题，之后提出新的模型来解决这一问题，并说明新的模型在性能上比现有方案更好，最后提出新模型的部署算法。

3.2　动态变化的租户带宽保障需求

3.2.1　典型的租户应用场景

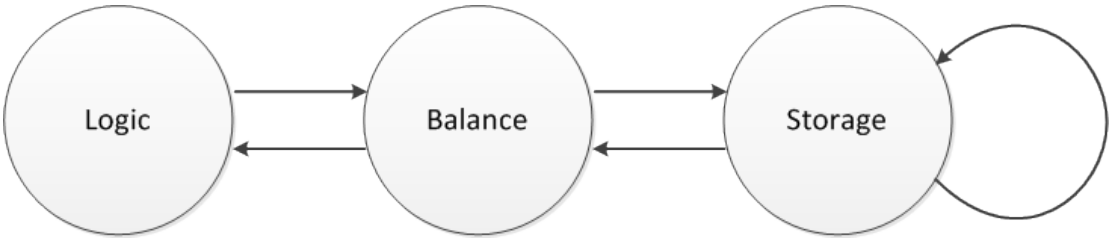


图 3.1　典型租户应用的通信结构图

租户在云上部署的虚拟机集群，通常会因为应用的性质而分为几个部分，每部分行使不同的功能，这些不同的部分称为组件。图3.1中表现了一种常见的租户应用通信结构，这种应用场景常见于各种网站的后端系统，如各大网站经常使用的Redis缓存系统[23]（以及其分布式扩展Codis[24]），它们都需要部署在多台服务器上，而在使用云服务时，就是由多台虚拟机组成的一个租户网络。这个租户应用分为三个组件，分别是逻辑、负载均衡和存储组件，每一个组件都是由一些虚拟机组成。图中组件间的边表示组件间虚拟机的通信，存储组件指向自身的边表示存储组件内部虚拟机间也存在相互通信。

在Codis这类应用场景中，租户有大量数据存储在存储组件的虚拟机中，逻辑组件中的应用需要用到这些数据时，会向负载均衡组件请求数据，负载均衡组件根据一定原则从存储组件取得数据并递交给逻辑组件。组件间的通信速率直接影响了逻辑组件获得数据的时间，而逻辑组件是网站的后台，网站用户访

问网站时会向其请求数据，所以组件间通信速率决定了网站用户访问网站的速度。如果这些组件间通信因为受到云数据中心中网络性能不稳定的影响而发生拥塞，就会使得网站用户访问延迟增加，严重影响用户体验从而影响网站效益。如果能提供带宽保障，就能够有效缩短访问时间的最大延迟，使得网站的服务质量提高。

在存储组件中，由于存储的数据需要备份和同步，所以经常会有存储组件内部的通信。这些通信的速率决定了数据备份和同步的效率。在一个更新频繁的缓存系统中，数据备份和同步的速率也十分重要，所以租户也会需要这一组件内部通信的带宽保障。

在Codis这类应用中，随着网站规模扩大，要存储的内容逐渐增加，存储组件的虚拟机势必需要增加扩容，负载均衡组件也有可能需要增加虚拟机，而所有组件间的通信带宽也可能会增加。这时就会发生租户带宽保障需求的变化。

3.2.2 租户需求变化带来的问题

如上所述，实际应用中，租户租用的虚拟机集群是会动态变化的，随着业务增长，需要的计算资源会增加，租户请求的虚拟机数量就会增加，同时租户需要的带宽也会增加。

现有的带宽保障方案中，并没有考虑租户后续改变需求的情况，少数提及的方案中也只是简单的将新需求重新计算后重新分配虚拟机。而在实际中，重新分配虚拟机并不是一件简单的事，重新分配会导致部分虚拟机需要迁移，而对于Codis这类存储应用，迁移虚拟机将造成大量数据传输，不仅占用虚拟机槽位，也会占用网络资源，造成数据中心计算和网络资源都被大量消耗，会严重影响数据中心的性能。

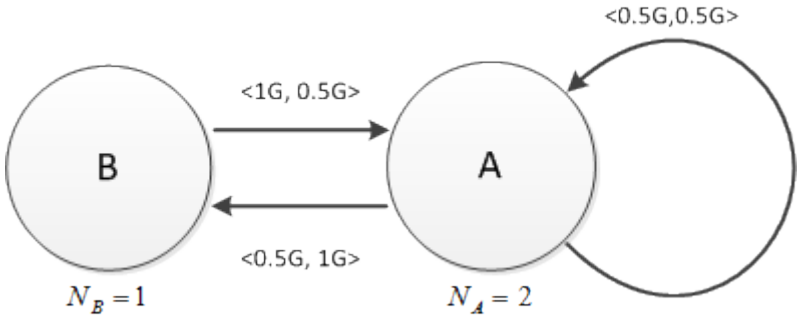
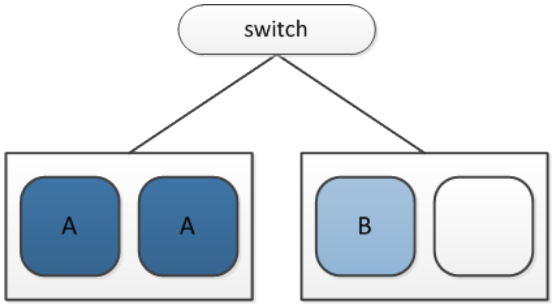
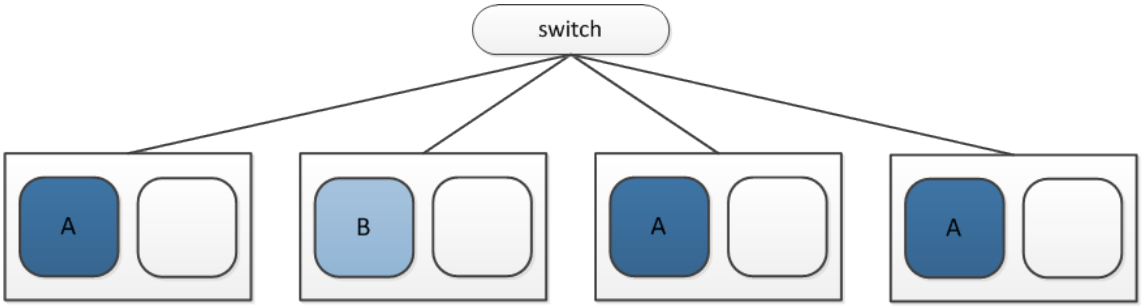


图 3.2　一个租户需求的TAG模型

图3.2所示为一个租户请求的TAG模型，这个租户的应用共包含2个组件A和B，组件A包含2个虚拟机，组件B包含1个虚拟机。其中A含有自指的边¡0.5G, 0.5G¿表示组件A内部虚拟机间需要0.5Gbps通信带宽保障。而组件A与B之间的边表示从A发送的流量有每个虚拟机0.5Gbps的带宽保障同时B能够以1Gbps每个虚拟机的速率接收；反过来B能够以1Gbps的速率发送而A能保障每个虚拟机以0.5Gbps的速率接收。注意由于A包含2个虚拟机所以AB间双向的吞吐率均是1Gbps。



(a) 初始分配结果



(b) 扩容后分配结果

图 3.3　TAG模型的虚拟机分配结果

图3.3(a)表示的是上述的TAG在云中的虚拟机分配情况，图中有2个物理机，每个物理机包含2个虚拟机槽位，每个物理机都有一个1Gbps的链路连接到交换机。第一个物理机分配了组件A的2个虚拟机，第二个物理机分配了组件B的1个虚拟机。AB间的1Gbps通信带宽占据了两条链路，而A内部的通信由于所有A的虚拟机在同一个物理机中，节省了这部分的带宽。

这时，由于租户的需求增加，组件A需要扩容，虚拟机数量从2上升为3。注意这时将组件B所在物理机空闲的槽位分配给这个新增加的虚拟机并不能满足租户的带宽保障需求。对于组件A中的每个虚拟机，组件内通信需要0.5Gbps带

宽，与组件B通信需要0.5Gbps带宽，共需要1Gbps带宽，而每个物理机的链路带宽是1Gbps。能够满足要求的分配方法如图3.3(b)所示，组件A中的任意两台虚拟机不能分配在同一个物理机中，否则对B通信需要1Gbps带宽，而对剩下的一个A的虚拟机还需要0.5Gbps带宽，共需要1.5Gbps超过了链路上限。在这个例子中，组件A的一台虚拟机必须要迁移。

为了避免迁移带来的负面影响，减少迁移造成的资源占用，在这个例子中，第一次分配虚拟机时，就应当将组件A的两台虚拟机分开分配到不同的物理机中。而现有方案中的分配算法，都会尽量将组件A的虚拟机分配在同一个物理机内，因为一般情况下并置（colocation）可以节省带宽占用，而现有研究中并没有考虑租户需求变化的情况。一个简单的解决方案是，让租户指定需要预留给未来扩容的虚拟机数量，用户可以提供一个新的参数——并置系数——来定义租户期望的未来组件扩容的程度。并置系数c定义为：

$$c = \frac{N}{N_{max}}$$

其中N是这次租户请求的虚拟机数量，N_{max}是租户需要的最大虚拟机数量。在分配虚拟机时，考虑并置系数后，预留更多槽位，就能在之后扩容时将预留的槽位分配给新的虚拟机。

而这种简单解决方案的缺陷也十分明显，首先租户并不清楚未来租用的集群的扩容情况，而更严重的是，这样的方案需要额外占用虚拟机槽位，减少了数据中心资源的利用率，并且一个固定的参数并不能适应于复杂变化的数据中心虚拟机分配情况。所以需要一种动态自适应的方案来解决这一个问题。

租户虚拟机数量扩大后造成的另一个问题，是占用带宽会随着虚拟机数量变化而变化。TAG研究中称由于该模型带宽保障是针对每个虚拟机的，在虚拟机数量变化后，每个虚拟机的带宽不变，所以能够适应这种变化。但这个结论是不成立的，下面举一个例子。仍然是图3.2中的模型，在组件A的虚拟机数量从2增加到3后，每个虚拟机的带宽保障是0.5Gbps，所以组件A总共的发送吞吐率是1.5Gbps，而由于组件B一共只有1Gbps的接收带宽，所以A的实际发送吞吐率只能有1Gbps。这时要么组件A的每个虚拟机无法达到租户预期的0.5Gbps带宽，要么需要相应增加组件B中每个虚拟机的接收带宽使得$S \cdot N_S = R \cdot N_R$，而增加带宽又会造成需要重新分配虚拟机的问题（在这个例子中B的带宽无法增加，但在其他情况下要增加带宽往往需要将原本并置的虚拟机分配到不同物理

机上以增加总带宽，从而造成迁移问题）。所以在这种情况下，TAG模型并不能很好的适应租户需求改变的情况。

造成这个问题的核心是因为TAG以及同类保障连接着VM的虚拟链路带宽的hose model模型都没能做到将计算资源与网络资源解耦合。这些模型中，对于每个组件，无论虚拟机数量变化，还是每个虚拟机的带宽保障变化，都会造成组件总的带宽吞吐率变化。这就导致当虚拟机数量改变时，同时也会影响占用的带宽，从而导致上述的问题。

本文提出的新模型，将计算与网络两种资源解耦合，租户在改变需求时，可以分开改变虚拟机数量和占用带宽，互相不会影响。这样的设计使得租户在用这种模型描述需求时更加直观，同时提供了灵活性使得租户能够很好地通过模型对不同资源进行权衡取舍，并且通过这种灵活性，巧妙地解决了租户需求扩容时带来的迁移问题而不用使用强行预留槽位的方法。

3.3 弹性带宽保障模型

3.3.1 弹性带宽保障模型定义

本文提出的模型称为弹性带宽保障模型（Elastic Bandwidth Guarantee，EBG）。EBG既保留了TAG的优点，能够描述复杂的租户应用，又考虑了租户需求会动态变化的因素，通过将计算资源与网络资源解耦合，解决了因租户需求扩张产生的问题。

EBG模型与TAG一样，是一个有向图，图中的节点表示租户应用的组件，边表示组件间的通信带宽保障需求。更准确的，EBG是一个图$G = (V, E)$，其中V是节点集合，E是边集合。对所有节点$v \in V$，有一个表示组件中VM数量的参数N。对所有有向边$e = (u, v)$，这条边从节点u指向节点v，表示从组件u 发送到组件v的流量的带宽保障需求。这个需求由一组参数$< S, P >$表示，S表示组件u发送到组件v的通信的聚合带宽保障，即组件u发送到组件 v的所有流量带宽集合T中，满足：

$$\sum_{t \in T} t \geq S$$

这里只定义发送方发送的带宽而没有提及接收方接收带宽，是因为EBG模型定义接收方接收带宽等于发送方发送带宽，这样的设计使租户对带宽保障有

更清晰的认识，而不会出现前面例子中的接收带宽与发送带宽不一致从而使得实际带宽不能达到租户期望的问题。

注意这样的聚合带宽保障并不能保障每一个虚拟机的最小带宽，在这个定义中，即使一个组件的流量达到了保障的带宽S，也有可能其中部分虚拟机的带宽很低，甚至是零。为了弥补这一点，EBG的边还有另一个参数P，定义了发送方每个虚拟机的最小带宽保障。接收方的接收带宽同样由接收带宽等于发送方发送带宽的定义计算得到，即接收方每个虚拟机最小带宽保障P_R满足：

$$P_R = \frac{P_S \cdot N_S}{N_R}$$

而EBG中自指的边与TAG的定义相同，表示组件内部通信的带宽保障，而其参数为$< NP, N >$，表示组件内部通信中每个虚拟机的最小带宽保障P。

这样由总带宽保障、每个虚拟机带宽保障两个参数确定了一个带宽保障需求，而当一条边的$S = P \cdot N$时，EBG的这条边就等价于一个接收带宽等于发送带宽的TAG边，所以可以通过EBG模型描述一个TAG模型。

3.3.2 弹性带宽保障模型示例

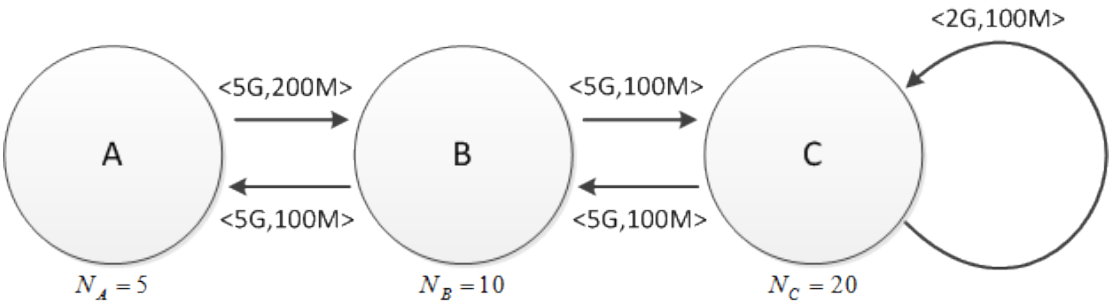


图 3.4 一个EBG的例子

图3.4是一个示例的EBG模型，包含三个组件。其中组件A到组件B的边¡5G,200M¿表示从组件A发送到组件B的总带宽保障是5Gbps，A中每个VM有最小200Mbps的带宽保障。组件C包含一个自指的边，表示组件C内部VM间通信有100M/VM的带宽保障。注意在EBG中有$S \geq N \cdot P$，且当等号成立时等价于TAG模型中的边，自指的边S参数没有意义且一直满足$S = N \cdot P$。

3.3.3 弹性带宽保障模型的优势

EBG模型的优点，除了与TAG类似便于表示复杂的租户应用场景以外，还通过提供灵活性解决了租户需求变化的问题。在EBG中，由于保障的是组件间聚合带宽，根据这个定义，无论组件中的虚拟机数量如何变化，都不会影响到带宽保障，同样改变组件间聚合带宽保障也不会影响到每个虚拟机的带宽保障，这就是EBG解耦合计算资源和网络资源的方式。租户改变需求增加虚拟机数量时，EBG模型的带宽保障在一定程度上不受影响，而避免了前面所述的迁移等问题。

EBG保障的聚合带宽，除去每个VM的带宽保障占用的部分，剩余的带宽称为弹性带宽E：

$$E = S - N \cdot P \tag{3-1}$$

弹性带宽E的存在，为EBG模型提供了灵活性，这部分带宽并不是固定分配给某一部分虚拟机，而是动态调整的，当租户的一部分虚拟机使用的带宽较少时，同组件中其他虚拟机可以使用这部分带宽。这种灵活性使得EBG模型能够更加节省数据中心的网络资源。下面举一例子。

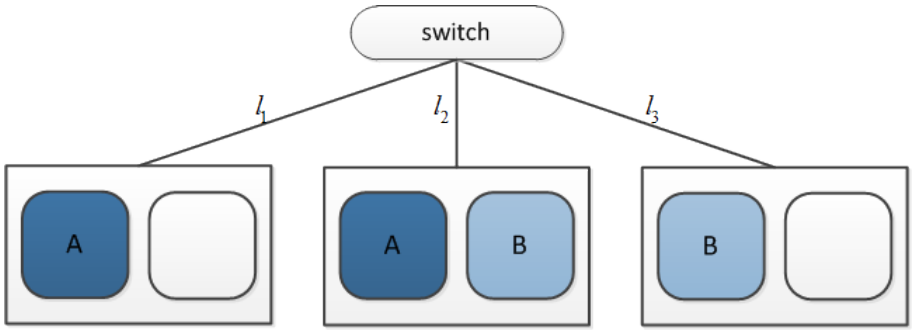


图 3.5　EBG模型能更加有效的利用带宽资源

图3.5中有3台物理机，每台有一条1Gbps的链路，两个VM槽位。现有两个组件A和B，都含有2个VM，这两个组件都需要对外发送流量，如果使用TAG模型，假设A和B总对外带宽各需要有1.5Gbps，每个VM的带宽保障是750Mbps，那么会发现，在图3.5的拓扑中，无法分配虚拟机以满足带宽保障的需求，只有将每个VM的带宽保障需求减少到500Mbps，才能找到符合的分配方法，而这时两个组件各自只有1Gbps的带宽。

而若使用EBG模型，租户只请求组件聚合的带宽保障需求，A和B各1.5Gbps，由于一共3条物理链路共3Gbps带宽，能够满足要求，EBG的分配算法能够找到分配方法满足租户的需求。而实际如图3.5分配后，虽然第二台物理机上的两个虚拟机同时以最大速率发送数据时，每个只能得到500Mbps的带宽保障，但当A或B组件之一使用的带宽减少时，这两台虚拟机可以获得更大带宽，在最好情况下可以达到最大的1Gbps带宽。当然租户如果需要每台虚拟机的最低带宽保障，可以通过设置*P*参数来实现。

而*S*参数与*P*参数的关系，即弹性带宽*E*的大小，决定了租户增加虚拟机数量会产生迁移的概率。理论上，按照EBG的分配算法分配虚拟机后，一个组件的虚拟机数量可增加到而不发生迁移的最大值是（假设组件只有一条非自指的边，且没有其他组件影响）：

$$N_{max} = \frac{S}{P} \tag{3-2}$$

这个结论十分直观，图3.5的例子中，由于没有*P*参数的需求限制，假设虚拟机槽位没有限制，只要总聚合带宽不变，无论如何增加虚拟机，都不会对带宽保障产生影响。而当有*P*的限制时，相当于组件一共占用了*S*的带宽，而这些带宽会被每台虚拟机使用掉*P*，那么总共能容纳的虚拟机数量就是由式3-2所确定了。

如果租户希望在扩容后不发生迁移，那么可以在请求虚拟机时，选择足够大的弹性带宽。相比于设置并置系数，弹性带宽都是可以为租户所使用，而不像并置系数的方案中，只是单纯浪费了槽位资源。在3.2.2小节的例子中，如果使用EBG模型，租户请求¡S=1.5Gbps,P=0.5Gbps¿的带宽，那么在第一次分配时就会将两个VM分配到不同的物理机，而之后虚拟机数量增加到3时，并不会发生迁移。

EBG模型通过弹性带宽，提供了灵活性，使得在分配给租户资源时能够更加节约，降低了云服务供应商的成本。同时，租户可以通过选择合适的总带宽和每虚拟机带宽，来解决扩容带来的迁移问题，通过选择更高的总聚合带宽来获得更好的扩展性。

3.4 弹性带宽保障模型的部署

EBG是一个抽象描述租户需求的模型，需要实际部署方案来运用这个模型。一般的租户带宽保障方案中，云供应商将租户的需求模型作为输入，通过虚拟机分配算法，进行准入控制（admission control），检测租户的需求能否被满足，之后将云数据中心中合适的虚拟机分配给租户，并根据带宽保障方案预留带宽（一般的方案中是通过预留带宽实现带宽保障，而[14,15]等研究中则是通过速率限制来实现带宽保障）。

3.4.1 组件间聚合带宽保障的实现

TAG以及hose model等模型的虚拟机分配算法中，虚拟机数量和占用带宽完全对应，所以可以简单的计算得到能够在一个数据中心物理网络中分配的虚拟机。而EBG模型中弹性带宽部分并没有明确的与某部分虚拟机挂钩，所以分配虚拟机时的方法要特别考虑。本文设计了链路组（Link Group）算法，来实现弹性带宽的保障。

一个链路组(L, B)是由一组链路的集合L和所要保障的带宽B组成，表示这一组链路的聚合带宽保障是B。用链路组可以表示一个EBG的弹性带宽保障需求。图3.5的例子中，组件A的带宽需求是¡S=1.5Gbps,P=0¿，则对应的链路组是$(\{l_1, l_2\}, 1.5G)$，表示链路l_1和l_2一共需要1.5Gbps带宽；同样的组件B的带宽需求对应于链路组$(\{l_2, l_3\}, 1.5G)$，表示l_2和l_3共需要1.5Gbps带宽。

验证链路的剩余带宽满足链路组的要求的条件是，对于链路l以及所有包含链路l的链路组$G_i = (L_i, B_i), l \in L_i$，满足：

$$\sum_{l_i \in L} R_{l_i} \geq \sum_i B_i, \quad L = \bigcup_i L_i \tag{3-3}$$

其中R_{l_i}表示链路l_i的剩余带宽。上式表示对于一条链路l，当其所涉及的所有链路组需求的带宽总和不超过这些链路组包含的所有链路的剩余带宽总和时，这条链路上的链路组需求就能被满足。

图3.5的例子中，对于链路l_1和l_3，只涉及一个链路组，需求带宽1.5G，而包含的2条链路共剩余带宽2G，满足条件；对于链路l_2，组件A和B的两个链路组共需求带宽3G，而涉及到的所有3条链路的剩余带宽共3G，也满足条件。所以图3.5中的聚合带宽保障需求被满足，能够分配虚拟机给租户。

对于存在每虚拟机带宽保障要求的租户请求，要先将链路带宽中扣除这一部分带宽，剩下的带宽是计算上式判断条件的剩余带宽，而链路组需求的带宽也同样是扣除了这一部分带宽后的弹性带宽部分。

给定EBG模型中的一条边（非自指），以及这条边连接的两个组件的虚拟机在数据中心中的物理位置，就可以确定这条边对应的链路组。然而计算两个组件间路径并分层确定路径上的链路组计算量很大，而数据中心中绝大部分的拥塞发生在最低层的接入交换机链路上[7,15]，所以实际的算法中，只考察直接连接物理机的链路以简化算法。

有了上述判断条件，就有了验证EBG中弹性带宽保障的方法，从而可以实现EBG模型的虚拟机分配算法，提供组件聚合带宽保障。

大部分数据中心网络是类树形拓扑，本文将以单根树形拓扑的数据中心作为研究对象设计部署算法，而这一算法可以很容易的扩展到其他类树形拓扑上。

同TAG等研究一样，虚拟机分配算法的目的是根据租户的需求，在数据中心中找到合适的虚拟机槽位，满足租户对虚拟机数量和带宽保障的要求，如果不能找到，则返回告知搜索失败。暴力的搜索算法需要遍历所有虚拟机槽位组合，之后通过带宽保障的验证算法检验是否满足带宽保障条件。而由于一般将互相通信的虚拟机放置在同一个物理机或低层子树中可以节省带宽，所以同类研究的分配算法都采取将同一组件中虚拟机尽可能并置的类启发式搜索算法。

EBG的搜索算法伪代码如下：

```
1  def allocateEBG(tree, g):
2      sortComponent(g)
3      for component in g:
4          for node in tree bottom up:
5              if node.slots > g.N:
6                  allocate(component, node)
7              if success:
8                  # allocate next component
9                  break
10             else:
11                 deallocate()
```

```
12          else:
13              # searched all nodes in tree
14              return False
15      return True
16  def allocate(c, node):
17      checkStaticBandwidth(c, node)
18      if fail:
19          return False
20      if node is not a PM:
21          splitComponentToAllocate(c, node)
22      checkLinkGroup(c, tree)
23      if success:
24          reserveSlotAndBand()
25          return True
26      else:
27          return False
28  def splitComponentToAllocate(c, n):
29      s,ch=splitIntoChildren(c, n.children)
30      for component in s and node in ch:
31          allocate(component, node)
32      checkLinkGroup(c, tree)
33      if all success:
34          return True
35      else:
36          reuturn False
```

算法采用贪心的启发式搜索算法，算法输入是租户的需求EBG模型g，以及当前数据中心的状态tree，包括虚拟机槽位情况、静态带宽保留情况、动态链路组分布情况。算法主函数是allocateEBG()，判断一个EBG请求能否放置到当前树中。

allocateEBG()函数首先将EBG模型g中所有组件进行排序，这一步排序就是

启发式算法的关键。算法将占用带宽多以及每台虚拟机占用带宽多的组件排在前，这样在之后的搜索中，能够先将部署相对更困难的组件先放入树中，而相对容易的组件后放入。

之后算法一个个的将组件放入树中，在第4行的循环中，从叶节点自底向上遍历整棵树，通过allocate()函数寻找合适的放置组件的位置。如果找到能够放置的子树，则继续部署下一个组件，如果遍历所有子树仍然不能寻找到合适的位置，则搜索失败返回（第14行）。

allocate()函数是判断能否将一个组件放入一个子树的函数，这个函数是一个递归函数。allocate()首先检查组件中静态的带宽保障需求能否被满足（第17行），检查方法与hose model、TAG一样，对于子树n，组件中的一条边$e =< S, R >$，检查：

$$min(N_{in} \cdot S, N_{out} \cdot R) \leq B_n \tag{3-4}$$

N_{in}是这条边发送方组件在子树内的虚拟机数量，N_{out}是接收方组件在子树外的虚拟机数量，B_n是这个子树对外通信的上行剩余带宽。不等式左边表示这个子树在这条边的需求下，需要保留的带宽大小。

注意，边$e =< S, R >$中两个参数在TAG模型中是租户给出的，而EBG模型需要通过租户给出的P参数进行计算。

allocate()函数检查静态带宽后，如果失败则返回，如果成功则继续执行，此时如果子树node是一个叶节点，即物理机时，直接进入下一步骤，否则将递归调用splitComponentToAllocate()函数，把组件component拆分成若干小的组件，分别放入node的子节点中（第21行）。

在子节点部署完返回后，或者当前子树是物理机时，这一整个组件都已经部署完，可以进行弹性带宽链路组的检查（第22行），检查方法如3.4.1节所述的式3-3。

在检查成功后，这一次分配虚拟机成功，将虚拟机槽位、静态带宽都相应预留好，并将链路组信息保存入树中，返回成功（第24行）。

splitComponentToAllocate()函数将组件c拆分，放入子树n的子节点中（第29行）。拆分的原则是，尽可能让更多虚拟机并置在同一子树中，但同时又要满足带宽保障的需求。拆分后得到两个列表s和 ch，s是组件拆分后的小组件列表，每个小组件包含原组件的一部分虚拟机，但边信息相同。将新得到

的组件递归调用allocate()函数放入子节点中，所有都成功部署后，这一个组件完整的部署完，可以进行弹性带宽的检测。检测成功则返回，若失败，则回退并检查另一种拆分方式，直到成功或搜索完毕。

整个搜索算法需要遍历整棵树，每次遍历中，拆分算法需要遍历所有拆分方案，但通过带宽保障条件的限制，可以进行剪枝。同时算法复杂度可以通过限制时间剪枝的方法来大幅降低。之后的评价部分的实验中发现绝大部分成功的搜索只需要毫秒级的计算时间，而失败的搜索比例很小且可以通过限制时间来限定时间复杂度。

3.5 小结

本文首次提出了租户需求变化对于租户带宽保障的影响这一问题，同时提出了新的租户带宽保障模型来解决这一问题。 EBG模型的优势在于其灵活性，而其灵活性源于对之前研究提出的模型进一步抽象，于是EBG模型对租户应用能够更加贴近。通过灵活性，EBG模型解决了租户需求变化的问题，同时由于灵活性，使得其能够更好的利用网络带宽资源。

便于使用的模型带来的是部署模型的困难，本文提出了EBG模型的部署算法，能够充分发挥EBG模型的优势，同时实现该算法也能够为后面的性能评价实验所使用。

第 4 章　基于云的测试流量生成系统

4.1　引言

本文实现了一个基于云的测试流量生成系统——DCTG，该系统是一个基于真实应用目的，部署于云上，通过虚拟化技术实现扩展性的典型租户应用。本章首先介绍系统的目标和设计，之后介绍系统的具体实现，最后介绍系统的性能。

4.2　DCTG系统目标和设计

4.2.1　系统的功能与目标

基于云的测试流量生成系统（DCTG），功能是对网络实体进行测试，生成用户指定的大规模流量，发送给被测单位，同时收集统计信息。利用IaaS云服务，将测试流量生成系统部署在云端，能够有效利用云的特点，通过添加虚拟机就能够对系统进行扩展，实现高可扩展性，这就是基于云的测试流量生成系统的设计目的。

传统的硬件测试系统价格昂贵，并且难以扩展，而一般的软件测试系统性能差且扩展能力有限，基于云的测试流量生成系统通过IaaS云技术，能够真正实现高可扩展，使用普通硬件减少成本的同时有着极高的性能。

系统的具体设计目标如下；

- 可扩展。能够容易的扩展，通过云技术方便的增加系统的性能，通过分布式的方案能够生成大规模的流量；
- 生成用户指定的流量。能够通过用户配置，生成指定的网络层和数据链路层流量，实现用户的测试目的；
- 准确的生成流量及统计。根据用户指定的配置，准确的生成指定规模的流量，同时对流量实时统计，给出准确的统计结果。

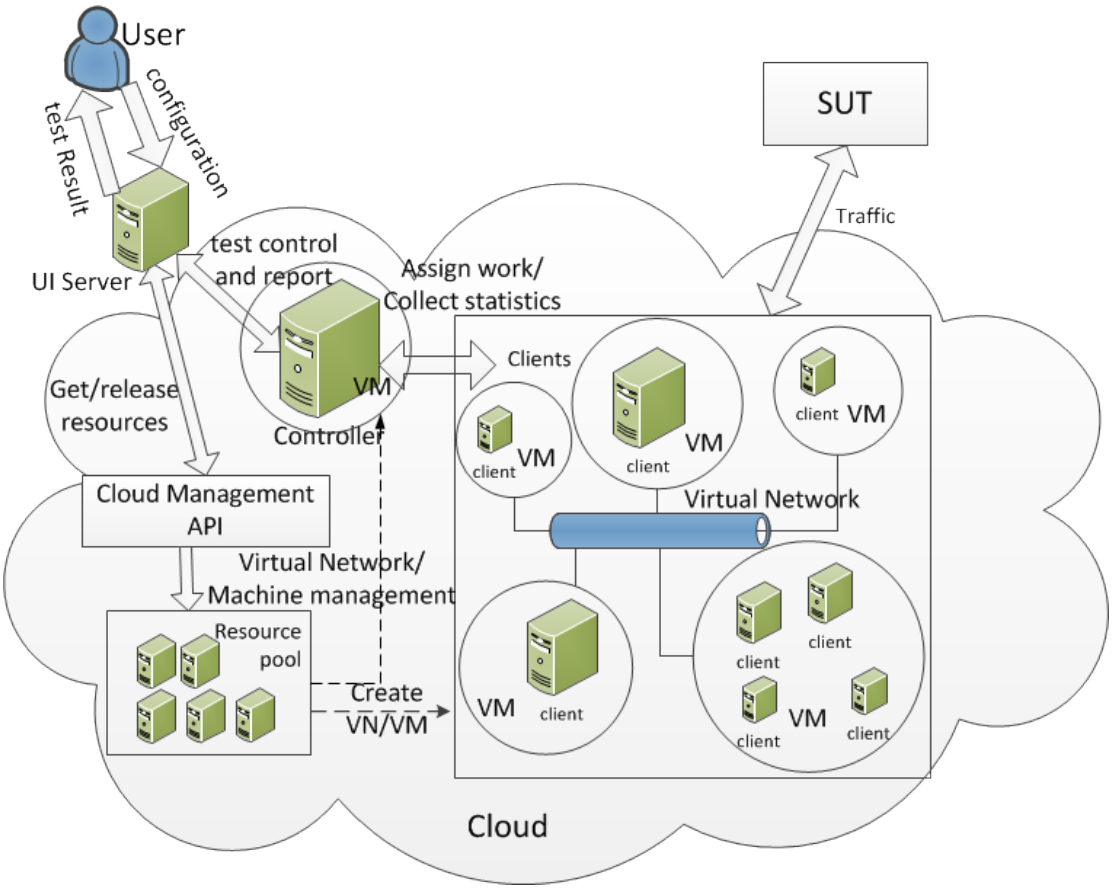


图 4.1　DCTG整体设计图

4.2.2　DCTG系统整体设计

DCTG系统是一个部署在云上的租户应用，为了达到设计目标实现高可扩展性的大规模流量生成，DCTG是一个分布式系统，通过将系统中的流量生成模块分布在多个虚拟机上，实现系统的可扩展性，同时通过大量的虚拟机并发生成流量来实现大规模的流量生成。

系统的整体设计如图4.1。整个系统部署在云上，由两个主要部分组成，分别是Controller和Client。

Controller负责读取用户配置、启动系统各个模块、配置硬件资源、发起client并分配任务、汇总client的统计。 Controller运行在单台虚拟机上，与所有的client进行通讯。用户将配置文件输入给controller之后， controller会启动相应的client生成测试流量。

Client负责流量的生成，并发运行在多个虚拟机上，向被测单位发送测试流量，同时统计测试数据并将统计结果发送给controller。

图中还表现了与云交互的部分，系统部署在OpenStack云上[25]，controller通过OpenStack API与云交互，请求虚拟机将系统部署在虚拟机上并建立虚拟网络来进行controller和系统其他部分的通信。

系统使用Erlang/OTP架构来实现[26]，利用Erlang语言的并发分布式运算能力和Erlang/OTP架构的基础模块来实现整个系统。 Erlang/OTP提供的模块能够有效地解决进程并发、错误处理、远程通信等分布式编程中出现的问题。

4.2.3 DCTG系统详细设计

4.2.3.1 系统工作流程

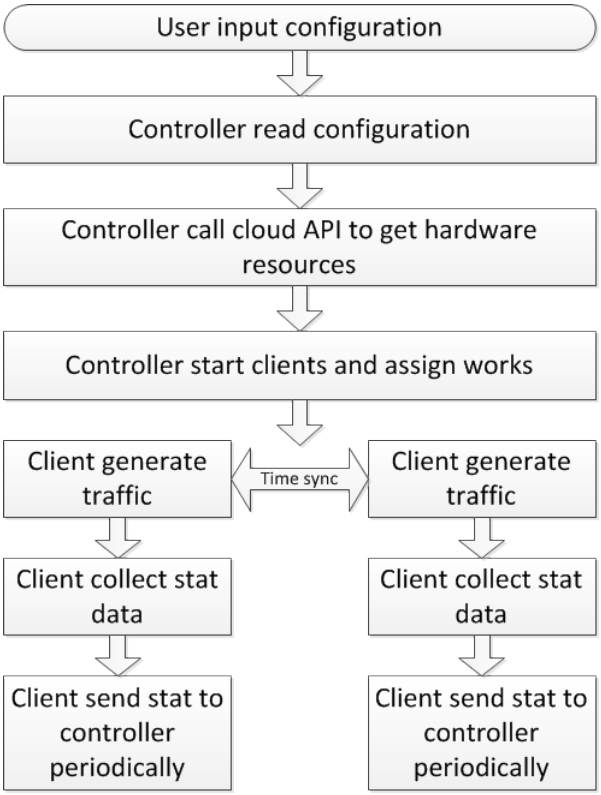


图 4.2 DCTG工作流程

如图4.2，DCTG系统的工作流程是，首先用户提供配置提交给controller，配置包括用户要生成的流量的规模和内容、被测单位的地址、测试持续的时间等。之后controller根据测试的规模，通过API调用云服务来获得测试所需要的虚拟机，并建立虚拟网络连接controller和这些虚拟机。完成后controller在这些虚拟机上启动client进程，并将用户指定的测试流量分配给各个client进程。被

启动的client进程得到测试的配置后，进行时间同步，并同时启动测试，生成流量，发送给被测单位。在生成流量的同时，持续实时统计流量信息，并发送给controller。controller 汇总后，将统计信息提交给用户。

4.2.3.2 系统模块设计

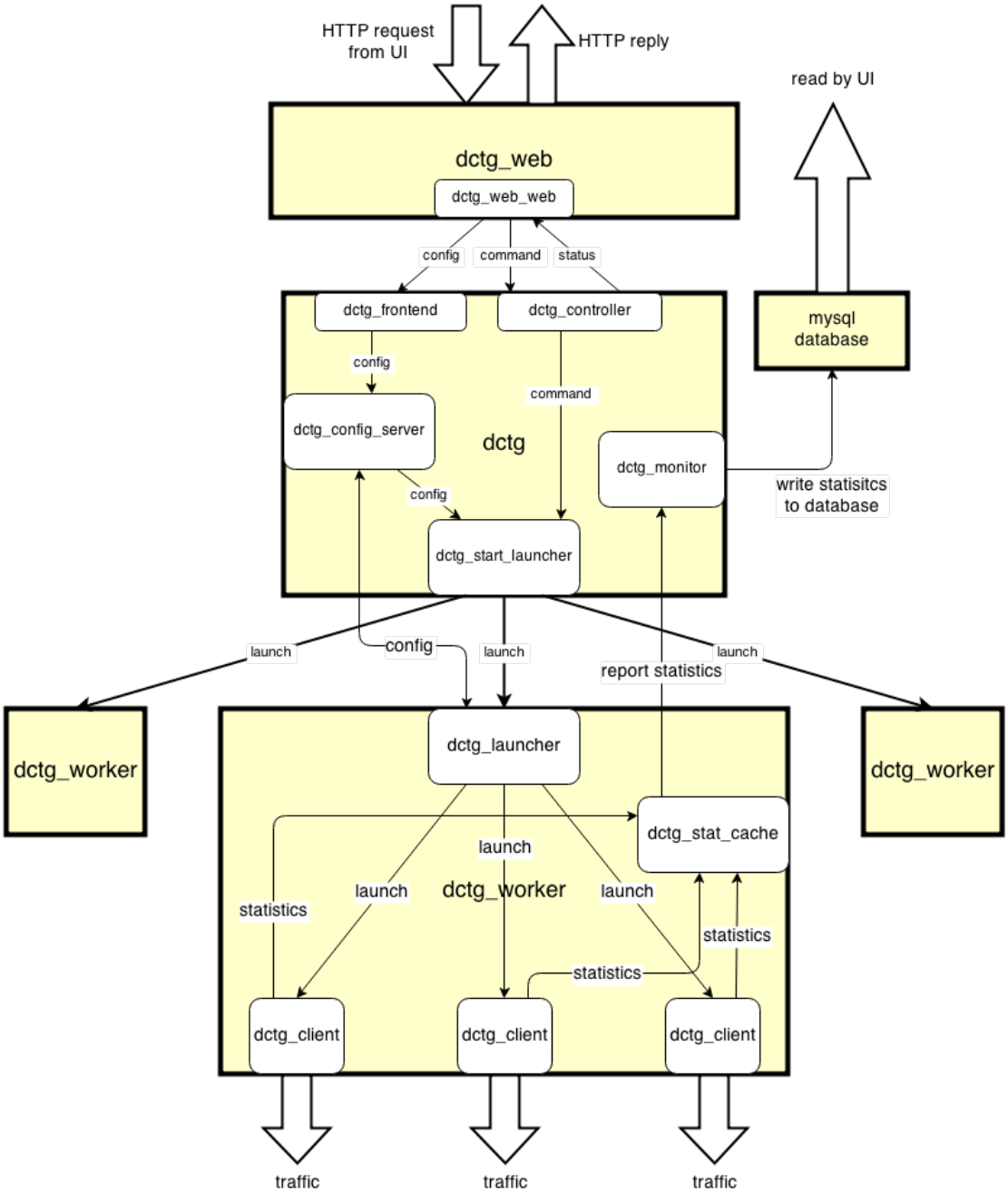


图 4.3 DCTG模块设计图

图4.3是DCTG系统的整体模块设计图。其中主要有三个模块：dctg_web、dctg、dctg_worker。 dctg_web模块的作用是与用户界面交互，接收用户的配置，并将系统信息反馈给用户。dctg模块即是整体设计中的controller，作用是分配任务、启动client、收集统计信息。dctg_worker就是工作模块client，生成流量并统计。下面详细介绍dctg与dctg_worker模块的具体实现。

4.2.3.3 详细模块设计

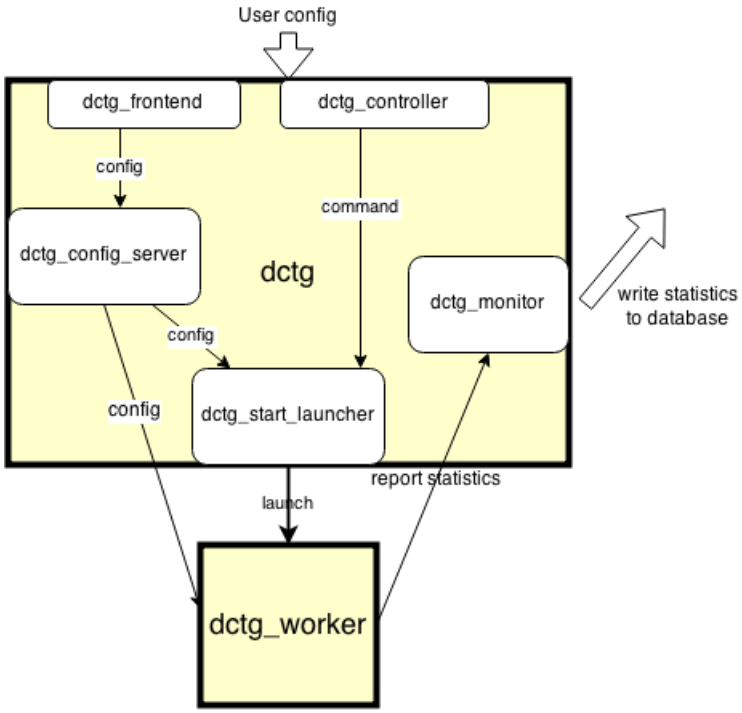


图 4.4 dctg模块设计图

图4.4是dctg模块的设计图。 dctg_frontend模块从dctg_web模块接收到用户配置，经过处理后（计算每个worker的任务）将其存入 dctg_config_server中。通过dctg_controller接收到用户的指令（开始、停止等），启动主要功能模块dctg_start_launcher来启动工作进程dctg_worker。启动工作进程时，从 dctg_config_server读出用户的配置发送给dctg_worker。dctg_monitor的作用是接收worker传来的统计信息，并进行汇总，写入数据库中供用户查看。

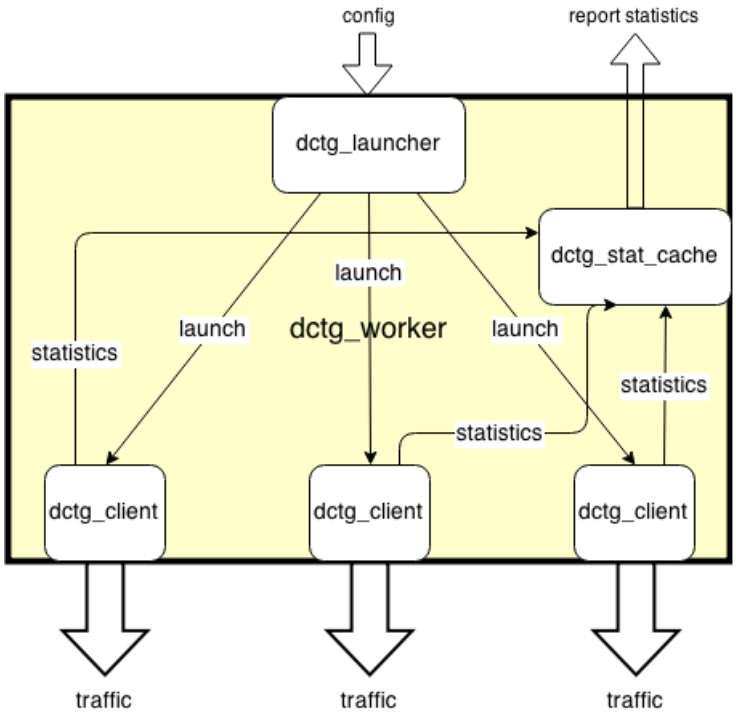


图 4.5　dctg_worker模块设计图

图4.5是dctg_worker的模块设计图。dctg_worker被启动后，从dctg_config_server中得到自身的任务及配置，等到dctg模块发出的同步开始命令后，依照配置开始生成流量。生成流量通过启动多个并发的dctg_client进程（erlang进程），来高速率的发送流量。

在发送流量的同时，每个dctg_client还进行实时的统计，并将统计通过dctg_stat_cache 汇总并发送给dctg模块。

4.2.4　几个要点

系统的设计能否达到要求，有几个要点需要论证。

4.2.4.1　系统可扩展性

系统的可扩展性，就是系统在扩大规模后仍然能够正常工作。需要考察在增加client规模时，系统能否达成设计目标。

当运行大量client时，可能对系统产生性能影响的环节包括：

1. 系统启动阶段。controller首先要启动所有client，并将配置下发。controller在启动client时是并发的，通过启动多个并发的进程，每个负

责一个client的启动，这样的情况下，总启动时间T需要：

$$T = \Delta_{thread} \cdot N + \max_i(t_i)$$

其中Δ_{thread}是每个进程启动的时间，N是要启动的client总数，$\max t_i$是所有client中最长的启动时间。由于Erlang中启动百万数量级的进程也只需要近似常数的时间[26]，所以总启动时间只被所有client中最慢的启动时间限制，而一般的启动最长也只需要秒数量级的时间，所以系统启动阶段完全具备可扩展性。

2. 统计阶段。client会实时进行统计信息收集，存储到dctg_stat_cache模块，之后发送给controller。为避免controller接收统计信息负载过重，client会间隔一段时间来发送统计信息，间隔时间设定为1秒，按照client数量为1000来记，每秒controller也只会收到1000个数据，完全能够承受，而1000个client一般已经能够生成极大规模的流量了。当系统规模更大时，需要采取层次化的统计收集方式，如图4.6，这样可以极大减少controller的负担。而目前还并不需要如此规模的client。

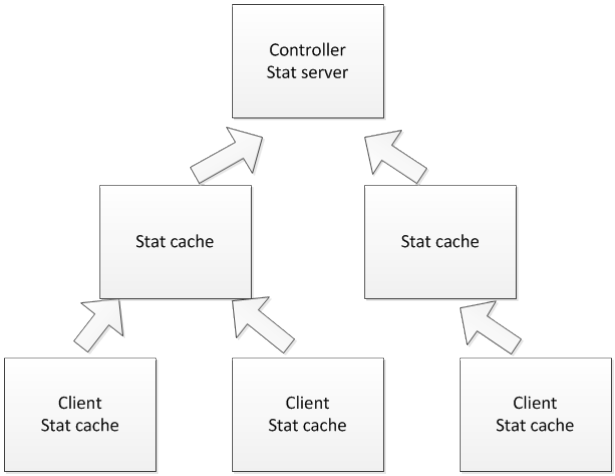


图 4.6　分层收集统计信息

由于除了启动阶段和统计阶段，controller与client之间并无交互，所以DCTG系统完全具备极高的可扩展性。

4.2.4.2　生成流量准确度

生成的流量规模能否完全依照用户的要求，就是流量生成的精确度。在大

时间尺度上看，生成的流量是能够符合要求的。但由于client间的时钟同步误差、系统软时钟或定时器的精度误差以及系统进程调度的时间误差，可能造成小尺度上流量的不稳定，这是通过软件生成流量必然存在的问题。

一方面，client之间存在的时钟同步误差，通过适当的同步机制可以尽量减小误差，并认为其导致不同client之间存在一个上限为δ的时钟相位偏移（δ极小，如NTP可实现ms级别同步）。假设client均匀产生流量，则该误差只影响测试起始阶段（例如第一个1s）和结束阶段（例如最后一个1s）生成的流量精度，对中间阶段生成的流量精度没有影响。

另一方面，系统时钟精度误差（以及相关的进程调度时间误差）会造成流量生成任务执行时间的误差，从而造成生成不均匀流量，精度主要由这个因素决定。在较新的Linux内核（2.6.16以上）上部署Erlang，可提供us级的时钟精度，其进程调度精度也远小于1ms，可以达到软实时效果（soft realtime）。

DCTG能够生成符合用户要求的准确的流量。

4.2.4.3 实时统计准确度

实时统计功能能否反应真实的流量生成情况，就是实时统计的准确度问题。

对于实时统计数据，实际上是一个$\{D, T\}$的二元组，表示在测试进行了时间T时相应统计量为D。统计数据的误差有两部分，数据D的误差指实际的数据与记录的数据的差异，而这在本系统中可以认为不存在；另一部分是T的误差，即产生数据D的真实时刻和记录时间T所对应的时刻之间的差异。由于需要统计多个client上的数据，其误差可由下面几个因素造成：

1. Client向controller发送统计数据到controller收到数据的时刻之间存在传输延迟，但不会影响数据的内容。不同的client先后发给controller数据，controller汇总时只需要等待收到所有数据再统计最终结果，不会导致误差。
2. Client的系统时钟与真实时刻之间的偏差，与4.2.4.2节的同步误差情况分析类似，可认为不同client之间经过时间同步，存在一个上限为δ的时钟相位偏移。同步误差影响了T，但是如果根据统计数据计算的最终结果是瞬时值而不是累加值，则该误差也只对测试起始阶段（例如第一个1s）和结束阶段（例如最后一个1s）的统计结果产生影响。

统计的误差在DCTG系统中十分小，与生成的大规模流量相比，可以忽略。

4.3 DCTG系统实现

4.3.1 重要功能算法

本节介绍DCTG系统中，重要功能的实现算法。

在4.2.3.3一节中，介绍了dctg_worker模块中dctg_launcher模块是用来按照用户配置，生成dctg_client模块发送流量的。dctg_launcher模块的启动dctg_client算法是DCTG系统按照用户配置准确生成流量的核心。

dctg_launcher模块精确达到用户要求的速率的方式是，设定一个规定的时间间隔（一般为10ms），根据用户要求的速率计算出这个时间间隔内要生成的dctg_client数量，按照时间间隔设置定时器，每次触发定时器就生成指定数量的dctg_client来发送流量。之后通过设置触发定时器的间隔尽可能的保证每次触发都按照规定的间隔进行，如果无法按时完成则设置时钟为0（立刻开始下一轮），并发出警报记录在log文件中（说明系统性能达到瓶颈）。

设置合适时钟的方法详细介绍如下：

每次计算的输入为开始时刻StartTime、每一次触发设定的间隔时间Interval、当前已经过了多少次触发Round、当前时刻CurrentTime。输出为下一次触发的间隔时间Timer：

$$Timer = StartTime - CurrentTime + Interval \times (Round + 1)$$

当$Timer < 0$时令$Timer = 0$。

举例说明，要求的速率为每秒建立1000个tcp连接，设定的间隔时间为10ms，则每次触发要生成10个dctg_client进程（每个dctg_client负责一个TCP连接），若某一次触发时开始时刻和当前时刻差为-98ms，已经经过了10轮，那么下一次触发时间应当为12ms后；若某一次触发时间经过为102ms，已经经过了10轮，那么下一次触发时间为8ms后；若时间经过为112ms，已经经过了10轮，下一次触发时间为0ms即立刻执行下一批生成10个dctg_client任务。

这样的算法，保证了DCTG系统生成流量的准确，同时让流量的生成十分平滑（10ms间隔计算发送的流量）。

4.3.2 系统物理部署

DCTG系统搭建在OpenStack云上，首先需要搭建OpenStack云，然后在云的虚拟机上启动DCTG系统。

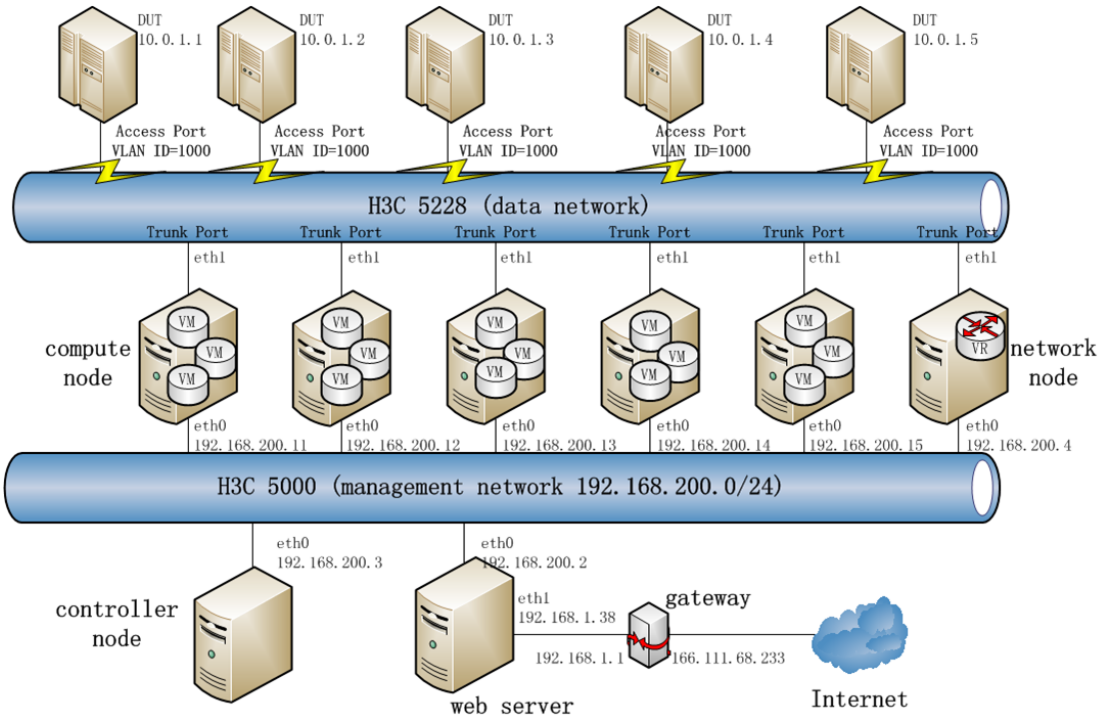


图 4.7 DCTG系统部署图

如图4.7，使用了8台服务器来搭建OpenStack云。其中，Controller node是OpenStack的控制节点服务器，Network node是OpenStack的网络控制节点服务器。 Web server是DCTG系统用来与用户交互用的前端界面所在服务器。其余5台服务器均为OpenStack 的计算节点服务器，用来启动虚拟机。

为了提高性能，分隔OpenStack通信和DCTG的通信，这些服务器通过两个交换机连接起来，组建了两个局域网。图中5000交换机连接的192.168.200.* 网段是OpenStack的管理网段，所有服务器通过这个网络进行OpenStack的各种控制信息的传输。而5228交换机则是数据网络，所有OpenStack管理下的虚拟机通过这个网络进行通信，而运行在其上的DCTG系统也是在这个网络上进行通信，并向这个网络上的被测单位发送流量。

4.4 DCTG系统性能

DCTG系统实现并部署后，进行了一系列测试，达到了设计目标，本节将对DCTG系统的性能进行介绍。

4.4.1 DCTG系统功能

本节先介绍DCTG系统的基本功能。

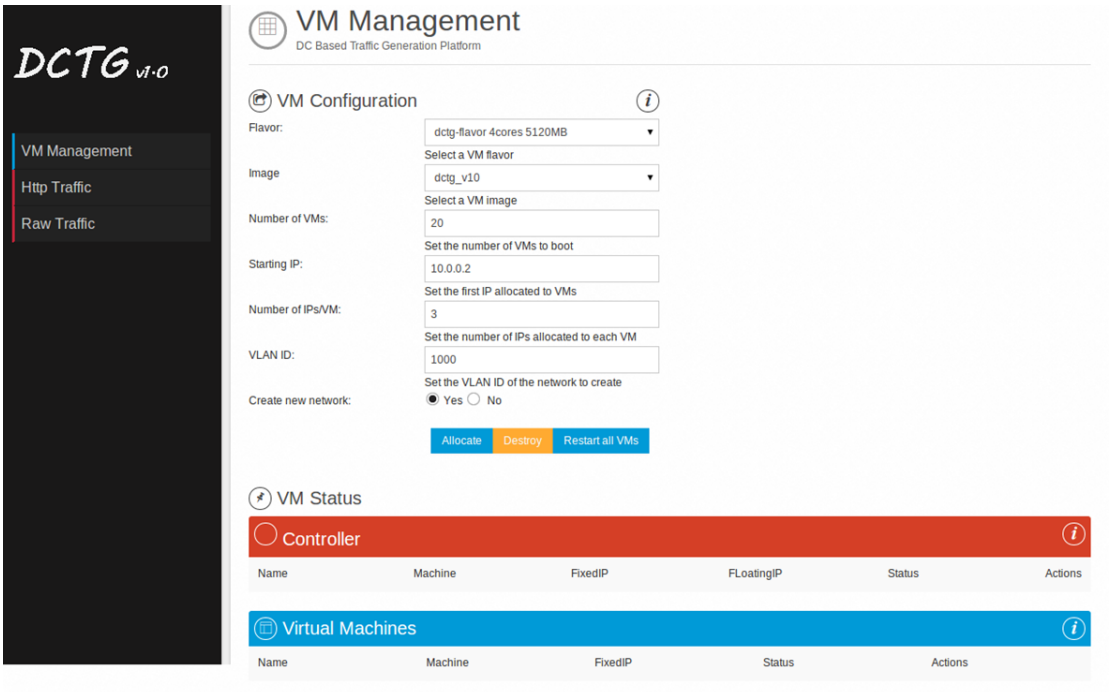


图 4.8 DCTG系统虚拟机管理界面

图4.8中是DCTG系统的虚拟机管理界面，该界面中可以进行虚拟机的启动、删除、重启等操作。 DCTG通过调用OpenStack API，来向OpenStack云请求虚拟机。

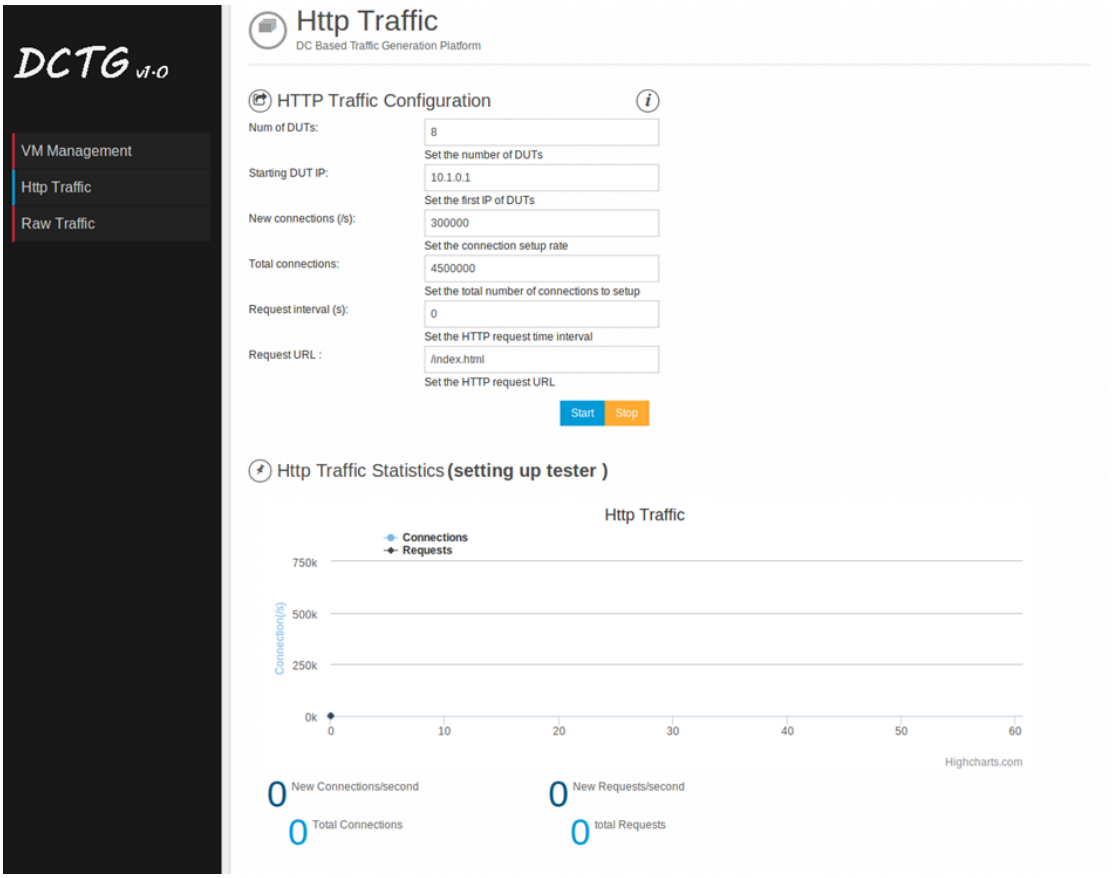


图 4.9　DCTG系统流量生成界面

图4.9是DCTG系统的测试流量生成界面，该界面中，可以配置要生成的流量，并对系统发出启动、停止命令，同时可以看到系统返回的统计信息。

4.4.2　DCTG系统性能

DCTG系统可以接受用户配置进行流量生成，为了测试系统的性能，使用了硬件测试仪作为系统发送流量的对象（即被测单元），通过硬件测试仪来准确的评价系统的性能。同时，在物理机上使用nmon等性能监控软件来观察DCTG系统对物理资源的使用情况。

测试使用了大量不同的配置，启动不同数量虚拟机（每台虚拟机的配置是4虚拟CPU，5G内存），进行了网络层、应用层、链路层流量的生成，通过DCTG系统自身的统计和测试仪的统计，对DCTG系统性能进行评价。

表4.1中是启动不同数量的虚拟机时，DCTG系统生成网络层及应用层流量时的性能，表中第二列数据是DCTG系统每秒新建TCP连接数，第三列数据是

表 4.1　不同虚拟机数量下DCTG系统的性能

VM数	TCP conn/s	HTTP request/s	memory
25	300k	800k	2.3G
20	240k	640k	2.3G
15	180k	480k	2.3G
10	120k	320k	2.3G
5	60k	160k	2.3G

每秒发送HTTP请求数，第四列数据是运行过程中每台虚拟机的内存消耗。从表中可以看到，在启动25台虚拟机时，可以与被测单位每秒建立30万个TCP连接，每秒发送80万个HTTP请求。而内存使用一直比较稳定，通过改变配置发现内存使用只与建立的并发TCP连接总数有关，并发连接越多使用内存越多。

可以看到，DCTG系统通过增加虚拟机，能够方便的扩展，而其流量生成能力是正比于虚拟机数量线性增长的，这说明DCTG系统具备良好的扩展能力，只要增加虚拟机资源，就能够获得更高的性能。

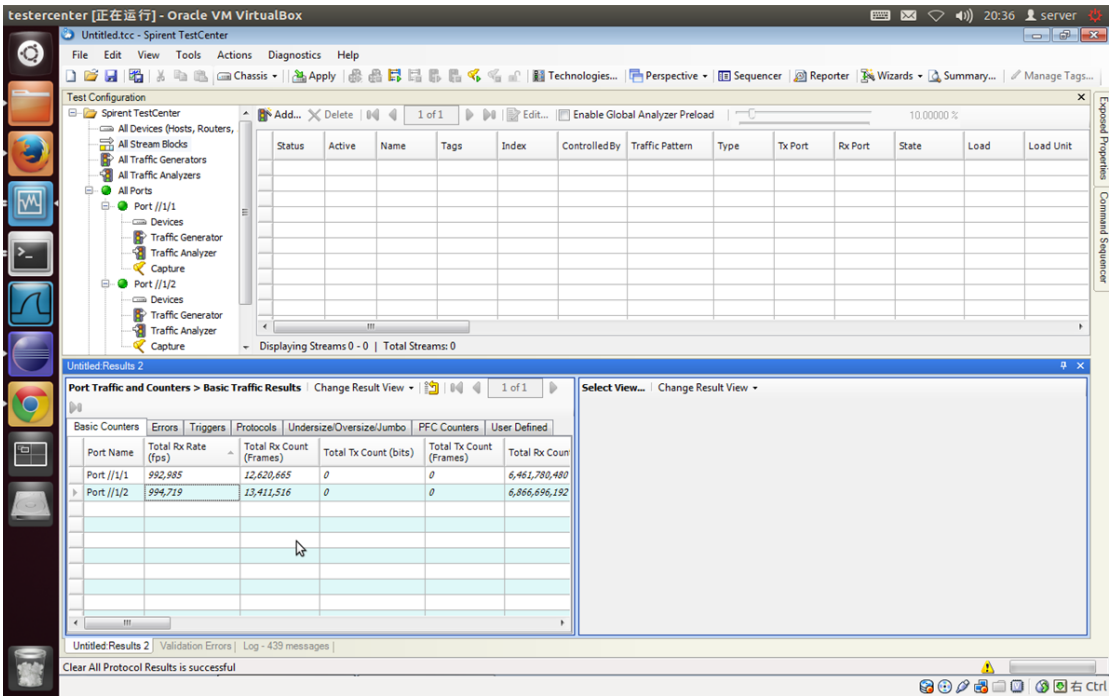


图 4.10　测试仪接收到2M pps链路层流量

测试还进行了链路层流量的生成，图4.10中是DCTG系统（共5台物理机）向测试仪（Spirent TestCenter SPT-9000A，两个1GE端口）的两个端口一共发

送 2M pps的报文（64Byte大小），测试仪显示两个端口各接收了 1M pps的流（64Byte报文大小）。也进行了不同大小报文的流量生成，在390Byte报文大小下，可以达到1.5M pps的速率，通过nmon软件可以看到每个物理机的网络端口（1000Mbps）都已被占满，而CPU和内存却没有怎么被消耗，说明DCTG系统的能力能够完全用满网络资源。

4.5 小结

DCTG系统的目标是实现一个高可扩展、高性能的能够准确生成用户指定内容和规模流量且具备准确实时统计功能的分布式测试流量生成系统。系统在设计上考虑了上述需求，并对其中部分重点问题进行了论证。在实现上依靠Erlang的高并发能力达成了系统的设计目标，并通过OpenStack云实现了方便的扩展性，能够通过添加虚拟机来获得更高的性能。实际部署后，对系统能力进行了测试，通过测试说明了系统能够生成大规模的流量。

DCTG系统是一个典型的IaaS云上的应用，并且通过增加虚拟机能够提高性能，作为一个典型租户应用，十分适合进行租户网络性能的相关研究，本文后面的部分会结合DCTG系统，进行租户网络带宽保障的研究。

第 5 章　模拟实验与性能测试

5.1　引言

前面的章节对租户网络性能的带宽保障问题进行了深入研究并提出了新的解决方案，同时设计实现了一个典型的租户应用，为了验证所提出方案的优势，本章将对其进行实验和性能评价。

本章首先对EBG模型进行了模拟实验，之后结合DCTG系统，进行了EBG模型的性能实验。实验结果说明了，EBG模型确实能够更加节省网络资源，同时能够解决租户需求扩张造成的问题。

5.2　EBG模型性能评价实验

5.2.1　实验目的和方法

为证明EBG模型的优势，将通过实验与其他模型进行对比，由于现有模型只有TAG能够表达应用的通信模式，所以至于TAG模型进行对比。模拟实验将从两方面说明EBG模型的优势。

其一是从租户请求接受率上。租户的请求并不是总能够被接受，而同样的资源，如果租户请求接受率更高，说明资源被更有效地利用了，该实验就是要通过租户请求接受率的比较，说明EBG模型能够更好的利用数据中心资源，从而为供应商节省成本；

其二是从租户请求扩容后，虚拟机迁移率上。根据前面第3章的论述，租户的需求会扩展，而这会造成问题，其中主要问题就是虚拟机迁移问题。迁移会带来严重的性能损耗，浪费大量数据中心资源，所以应当尽量避免。这个实验通过比较迁移率，来验证EBG模型通过配置更多弹性带宽，能够有效减少迁移。

为进行模拟实验，实现了EBG模型的部署算法，并将弹性带宽为零的配置作为TAG模型（此时EBG模型等价于TAG模型）输入，进行两种模型的对比。模拟实验在一个三层树形网络拓扑上进行，共包含2048台物理机，每个物理机

包含10个虚拟机槽位。网络上每个物理机的接入链路带宽是1Gbps，而聚合交换机和核心交换机的带宽分别为4Gbps和8Gbps。

5.2.2 接受率实验

实验中，EBG模型形如图3.4中的例子，因为这是最普遍的一种租户应用通信模式，并且包含模型中的全部功能。实验随机生成大量不同组件大小，不同带宽需求的请求。这些请求中虚拟机总数是均值为50符合指数分布的随机数，聚合带宽保障和每台虚拟机带宽保障取不同均值的指数分布来表示不同负载的请求（研究表明数据中心中租户虚拟机数量符合指数分布且均值为50左右，而租户的流量同样满足指数分布[10]），而作为对比的TAG模型也采用均值50的指数分布随机数作为虚拟机数量，总带宽需求均值与EBG模型相同，而每台虚拟机带宽需求则是平分总带宽需求。

实验在上一节所述网络拓扑中，生成300到500个租户请求，通过部署算法放入网络中，计算接受的请求与总请求数的比例。由于虚拟机槽位和网络带宽需求的限制，加上租户需求是随机生成的， 300到500个租户请求就可能将数据中心资源耗尽再也不能放入新的租户，所以实验时以连续3次部署失败作为资源耗尽的评判标准，达成后就不再生成新的需求，而是以当前请求数来计算接受率

实验比较了不同的弹性带宽下的EBG和TAG的接受率。 P/S 为随机生成的每虚拟机带宽保障均值与总聚合带宽保障均值的比值。租户请求中一条边的总带宽的均值设置为1Gbps、1.5Gbps以及2Gbps。随机生成300到500个请求后通过算法放入网络中，再计算接受率。实验结果如图5.1所示。

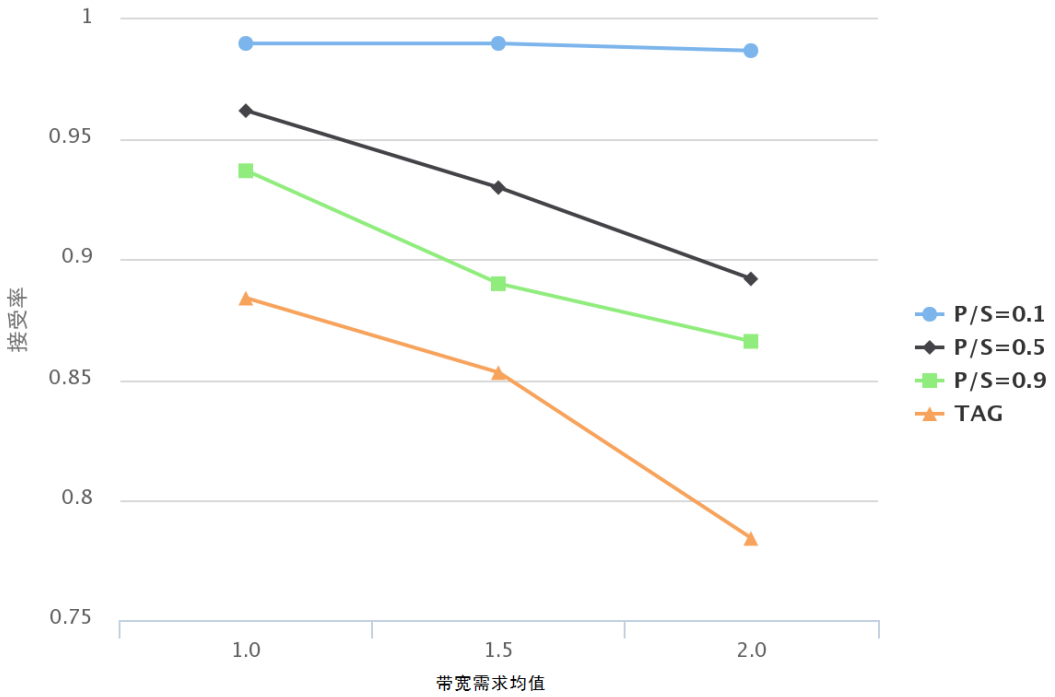


图 5.1　不同负载下EBG模型与TAG模型接受率比较

从图5.1中可以看到，随着带宽均值的提高，几种模型的接受率均有下降，而几种模型中TAG模型的接受率最低，并且在带宽均值提高时有着明显的降低。而EBG模型随带宽提高接受率的变化不明显，尤其当弹性带宽极高时（即P/S为0.1时）。这是因为EBG模型的灵活性使得数据中心网络的资源被更有效的利用了。

5.2.3　最大接受请求数实验

另一个实验能够从侧面体现EBG的接受能力更强。实验随机生成大量请求，并进行部署，直到连续3次分配算法失败返回，认为是数据中心网络已经接近饱和不能接受更多的租户请求，这时比较总共接受的请求数量。这个实验目的是比较不同模型在相同的虚拟机数量和总带宽需求下，整个数据中心网络能够接受的总请求数，能够接受更多请求的模型说明资源利用率更高。结果如表5.1所示，其中数据是多次实验取平均值的结果，所以请求数不是整数。

从表5.1中数据可以看到，即使增加P/S比例，EBG能够接受的请求数量并没有受到太大影响，而由于网络拓扑中总共有20480个虚拟机槽位，每个请求的

表 5.1　不同模型最大接受请求数比较

P/S	TAG	0.1	0.5	0.9
平均最大接受请求数	297.8	452.5	459.3	458.3

期望虚拟机数是50，所以这个最大请求数是受虚拟机槽位限制的。而TAG的接受能力却低了很多，且完全没有使用掉网络中的虚拟机槽位，可见是带宽资源限制了其接受请求。可以看到EBG模型的接受能力即使在每虚拟机带宽均值与总聚合带宽均值比值达0.9时，依然远超TAG模型，而这时不仅EBG模型请求的总带宽与TAG请求的总带宽相同，每个虚拟机的带宽保障也十分接近，EBG模型的带宽保障能力与TAG是几乎相同的。

从接受率实验中可以验证EBG模型的接受率高于TAG模型，而这正是由于弹性带宽的存在，使得EBG模型能够更好的利用数据中心的网络资源。

5.2.4　租户扩容后迁移率实验

这个实验先接受多个租户的请求，构成当前网络状态，记录下当前虚拟机的部署情况。之后随机将这些租户中的一部分的需求增加，增加其请求的虚拟机数量，之后计算旧有的虚拟机中改变位置的数量，比上虚拟机总数，得到迁移率。

实验首先在网络中部署100个虚拟机数量期望50的租户，之后按照一定比例增加虚拟机数量（在三个组件中随机选择一个），重新部署后，计算迁移的虚拟机数量与原虚拟机数量的比值。为比较TAG与EBG，租户的总带宽需求两种模型都是期望为1Gbps，而TAG的每虚拟机带宽需求按照定义由所有虚拟机平分总带宽，结果如图5.2所示。

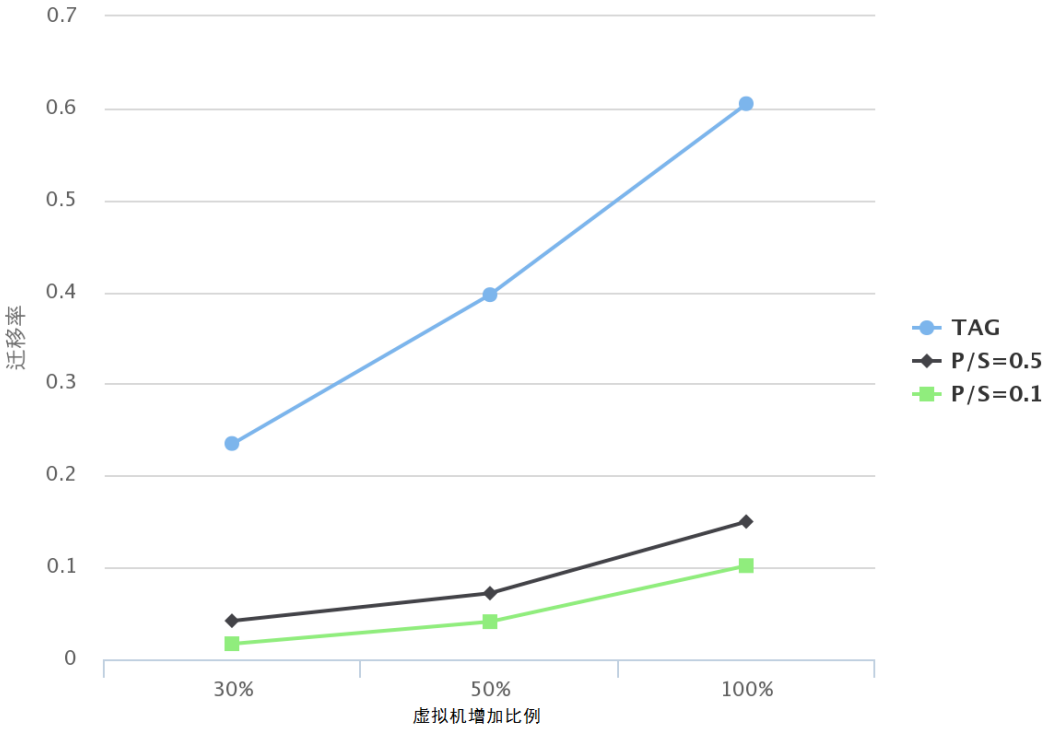


图 5.2　不同虚拟机数量增加比例下迁移率比较

从图中可以看到，虚拟机增加比例有30%、50%、100%三种，对于EBG模型来说无论P/S比例是0.1还是0.5，都只有很少一部分虚拟机发生迁移。而TAG模型却有大量虚拟机迁移，在虚拟机数量增加100%时，甚至有60%的原虚拟机发生迁移。

实验表明EBG模型确实能够通过充足的弹性带宽来减少发生迁移的概率，而且比起没有考虑扩容问题的TAG模型，EBG模型在这一问题上有着明显的优势。

5.3　结合DCTG系统进行EBG模型性能实验

5.3.1　DCTG系统建模

DCTG系统作为一个对网络性能十分看重的租户应用，租户带宽保障方案十分适用于这个系统。而在各类租户带宽保障模型中，EBG模型对DCTG系统尤其适用。因为对于DCTG系统而言，只关注总带宽，而不关心每个client 的带宽，如果总带宽足够高，就能生成需要的流量，而EBG模型保障租户组件的总

带宽的设计刚好符合这一点。 DCTG这样的租户应用在使用EBG模型时，能够充分享受到弹性带宽带来的优势。

要对DCTG系统使用带宽保障方案，首先需要对DCTG这一租户应用建模。使用EBG对DCTG建模如图5.3。

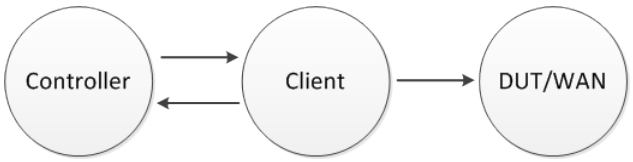


图 5.3　使用EBG对DCTG建模

DCTG系统包含两个组件，controller和client，互相之间有着通信，而一般这部分的通信主要是DCTG系统的统计信息从client 汇报给controller，并不需要很大的带宽保障。另外最主要的就是从client发送出去的流量，这部分流量会发送给被测单元。而如果DCTG测试的是云外的设备，这部分流量在云内部首先需要送到云的对外网关。所以可以看到，图中模型的最右边的组件有可能是云内部的被测单元，也有可能是出口网关。

这个EBG模型有一个特殊之处，就是其中被测单元（或者网关）这部分并不属于租户本身，这部分并不属于租户请求，而其在数据中心中的位置也已经确定。这样对于EBG模型的虚拟机分配算法，需要稍作改动，将这一部分看作是已经确定好位置的组件，无需进行分配，但需要考虑其带宽保障。

5.3.2　实验结果

将DCTG建模后，就可以进行实验。实验同样对EBG模型和TAG模型，比较其接受率和迁移率。实验中controller的组件平均大小为10VM，client的大小为平均30VM，被测单位大小平均为10VM，随机在云中选择位置。Controller与client间的平均带宽均为500，client发送到被测单元的带宽取不同的均值来进行对比。结果如图5.4。

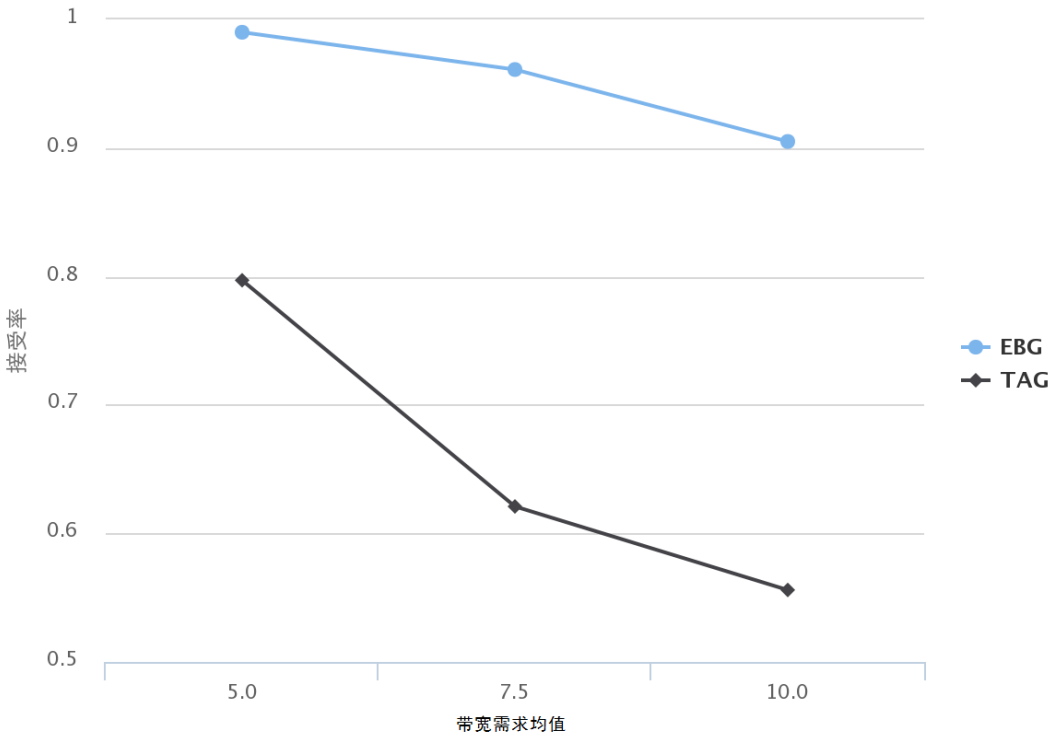


图 5.4　DCTG建模的接受率比较

图中可以看到，对于DCTG使用EBG和TAG分别建模，其接受率差异明显，EBG依靠弹性带宽几乎达到了100%的接受率，而少数不能接受的租户请求是因为被测单元的带宽限制。而TAG则接受率明显比EBG低。

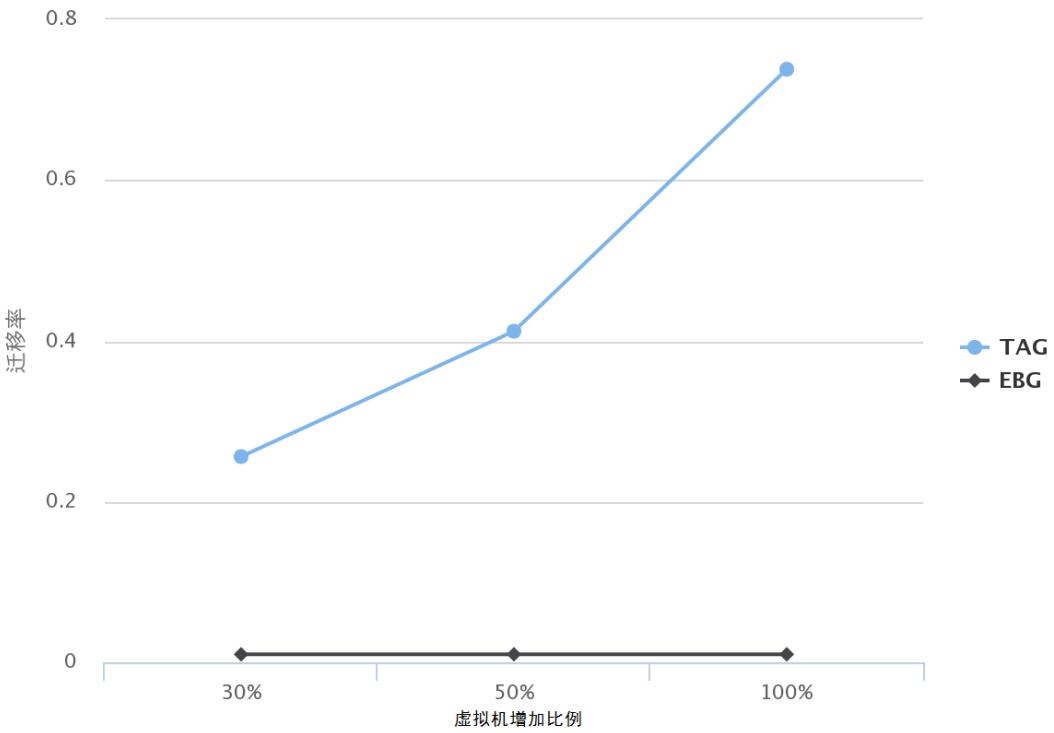


图 5.5　DCTG建模的迁移率比较

迁移率实验结果如图5.5。EBG模型几乎不发生迁移，而TAG模型仍然发生大量迁移。而这正是因为EBG模型建模DCTG系统，并没有每台虚拟机的带宽限制，拥有充足的弹性带宽。

由于DCTG应用的性质，使用EBG建模明显更具优势，而EBG模型的灵活性在DCTG这一类不需要每台虚拟机带宽限制的应用上有了最大的体现。实验结果完全说明，EBG模型拥有TAG模型所不具备的灵活性，同时能够解决租户需求变化带来的迁移问题。

5.4　小结

本章通过模拟实验，说明了EBG模型的性能优势。通过接受率的对比，说明了EBG模型能够更加有效的利用数据中心的资源；而通过迁移率的比较，说明了EBG模型能够有效解决这一问题。同时将DCTG系统建模，进行了实验并进一步说明了EBG模型的优势。

EBG模型的核心是弹性带宽和灵活性，通过灵活性，将租户带宽保障这一问题得以有效的解决。

第 6 章　总结

6.1　论文总结

IaaS云服务在如今愈发重要，大量企业用户由于现有的云服务还不能提供稳定的性能而无法放心将服务器部署在云上。提供带宽保障是让企业用户将服务器迁移到云上的重要一步，随着这项技术的发展和成熟，必然能够使得IaaS云服务也有着更大的发展。

现在已经有了大量关于租户带宽保障的研究，然而其中缺少对于租户需求扩展这一问题的深入探讨。然而，租户需求扩展是在使用云时必然且经常发生的情况，随着租户使用规模的扩大，这一问题发生会十分频繁。租户需求扩展会带来虚拟机迁移的问题，而对于常见的存储类租户应用，迁移是不能接受的。之前的租户带宽保障模型均没有能解决这一问题，所以本文提出了一种新的模型，通过巧妙的方法解决了这一问题。

本文的主要工作有：

1. 对租户带宽保障研究进行了综述，总结了当前已有的带宽保障方案，并介绍了当前主要的几种带宽保障模型。
2. 提出了租户需求扩展对于租户带宽保障问题产生的影响，说明了这一问题的严重性和常见性。
3. 提出了一种新的租户带宽保障模型EBG，通过模型的灵活性，既解决了迁移问题，又使新模型有了更好的性能，能够更有效的利用云资源。并设计实现了EBG模型的部署算法。
4. 设计实现了一个典型的租户应用，基于云的测试流量生成系统DCTG。介绍了系统的设计目标，给出了详细的设计方案来满足目标，并实现了系统进行了部署，对系统性能进行了测试说明了系统达成了设计目标。
5. 对EBG模型进行了模拟实验，通过实验说明了EBG模型的优势。同时利用EBG模型对DCTG系统进行了建模，并通过实验进一步说明了EBG模型的优势。

6.2 未来工作展望

未来的工作可以有几个方面，首先可以基于EBG模型实现一个真实的租户带宽保障方案，这一工作的难点主要在于结合真实的云数据中心（比如OpenStack）来实现准入控制、虚拟机分配功能，同时可以结合现有的work-conserving的带宽保障方案实现更加理想的租户带宽保障方案。

其次，可以进一步考察租户需求扩展对于带宽保障的影响，以及研究更好的租户需求扩展和虚拟机迁移算法，来提高租户网络的性能。

插图索引

表格索引

参考文献

[1] Shea R, Wang F, Wang H, et al. A deep investigation into network performance in virtual machine based cloud environments. INFOCOM, 2014 Proceedings IEEE. IEEE, 2014. 1285–1293

[2] Popa L, Kumar G, Chowdhury M, et al. Faircloud: sharing the network in cloud computing. Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication. ACM, 2012. 187–198

[3] Ballani H, Costa P, Karagiannis T, et al. Towards predictable datacenter networks. ACM SIGCOMM Computer Communication Review, volume 41. ACM, 2011. 242–253

[4] Ballani H, Jang K, Karagiannis T, et al. Chatty tenants and the cloud network sharing problem. NSDI, 2013. 171–184

[5] Xu Y, Musgrave Z, Noble B, et al. Bobtail: Avoiding long tails in the cloud. NSDI, 2013. 329–341

[6] Alizadeh M, Greenberg A, Maltz D A, et al. Data center tcp (dctcp). ACM SIGCOMM computer communication review, 2011, 41(4):63–74

[7] Kandula S, Sengupta S, Greenberg A, et al. The nature of data center traffic: measurements & analysis. Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference. ACM, 2009. 202–208

[8] Vasudevan V, Phanishayee A, Shah H, et al. Safe and effective fine-grained tcp retransmissions for datacenter communication. ACM SIGCOMM computer communication review, volume 39. ACM, 2009. 303–314

[9] Wang G, Ng T E. The impact of virtualization on network performance of amazon ec2 data center. INFOCOM, 2010 Proceedings IEEE. IEEE, 2010. 1–9

[10] Shieh A, Kandula S, Greenberg A G, et al. Sharing the data center network. NSDI, 2011

[11] Guo C, Lu G, Wang H J, et al. Secondnet: a data center network virtualization architecture with bandwidth guarantees. Proceedings of the 6th International COnference. ACM, 2010. 15

[12] Benson T, Akella A, Maltz D A. Network traffic characteristics of data centers in the wild. Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. ACM, 2010. 267–280

[13] Guo J, Liu F, Huang X, et al. On efficient bandwidth allocation for traffic variability in datacenters. INFOCOM, 2014 Proceedings IEEE. IEEE, 2014. 1572–1580

[14] Popa L, Yalagandula P, Banerjee S, et al. Elasticswitch: practical work-conserving bandwidth guarantees for cloud computing. ACM SIGCOMM Computer Communication Review, volume 43. ACM, 2013. 351–362

[15] Jeyakumar V, Alizadeh M, Mazieres D, et al. Eyeq: practical network performance isolation at the edge. REM, 2013, 1005(A1):A2

[16] Rodrigues H, Santos J R, Turner Y, et al. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. WIOV, 2011

[17] Crowcroft J, Oechslin P. Differentiated end-to-end internet services using a weighted proportional fair sharing tcp. ACM SIGCOMM Computer Communication Review, 1998, 28(3):53–69

[18] Yu L, Shen H. Bandwidth guarantee under demand uncertainty in multi-tenant clouds. Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on. IEEE, 2014. 258–267

[19] Xie D, Ding N, Hu Y C, et al. The only constant is change: incorporating time-varying network reservations in data centers. ACM SIGCOMM Computer Communication Review, 2012, 42(4):199–210

[20] Meng X, Pappas V, Zhang L. Improving the scalability of data center networks with traffic-aware virtual machine placement. INFOCOM, 2010 Proceedings IEEE. IEEE, 2010. 1–9

[21] Benson T, Akella A, Shaikh A, et al. Cloudnaas: a cloud networking platform for enterprise applications. Proceedings of the 2nd ACM Symposium on Cloud Computing. ACM, 2011. 8

[22] Lee J, Turner Y, Lee M, et al. Application-driven bandwidth guarantees in datacenters. Proceedings of the 2014 ACM conference on SIGCOMM. ACM, 2014. 467–478

[23] Redis. http://redis.io/

[24] Codis - yet another fast distributed solution for redis. https://github.com/wandoulabs/codis

[25] Openstack - open source software for creating private and public clouds. https://www.openstack.org/

[26] Erlang. http://www.erlang.org/

致　谢

感谢导师尹霞老师对我的指导和帮助，尹霞老师不仅在学术上，也在生活上给了我很多帮助和指导，而作为导师她也在平时的生活工作作风中给我们起到了榜样，同时对我遇到的各种困难和问题都给出了极大的帮助。

感谢施新刚老师对我一直以来的辅导，施新刚老师每周抽出时间与我进行讨论，并一直在学术研究上对我进行教导，让我学会了科学的思考方式以及学术科研的方法。

感谢张晗同学在研究中对我进行的帮助，张晗同学编写了十分美观方便的界面是我的系统能够使用。

感谢王之梁老师、余家傲老师以及向阳、姚姜源、吴丹、耿海军、孔祥欣、田庚、王太红、郭迎亚等实验室同学们对我的帮助和照顾，实验室的老师同学的学术氛围也一直影响着我，使我们共同进步。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

附录 A　外文资料原文

As one of the most widely used techniques in operations research, mathematical programming is defined as a means of maximizing a quantity known as objective function, subject to a set of constraints represented by equations and inequalities. Some known subtopics of mathematical programming are linear programming, nonlinear programming, multiobjective programming, goal programming, dynamic programming, and multilevel programming[1].

It is impossible to cover in a single chapter every concept of mathematical programming. This chapter introduces only the basic concepts and techniques of mathematical programming such that readers gain an understanding of them throughout the book[2,3].

A.1　Single-Objective Programming

The general form of single-objective programming (SOP) is written as follows,

$$\begin{cases} \max \ \ f(x) \\ \text{subject to:} \\ \qquad g_j(x) \le 0, \quad j = 1, 2, \cdots, p \end{cases} \tag{123}$$

which maximizes a real-valued function f of $x = (x_1, x_2, \cdots, x_n)$ subject to a set of constraints.

Definition A.1：　In SOP, we call x a decision vector, and $x_1, x_2, \cdots, x_n$ decision variables. The function f is called the objective function. The set

$$S = \left\{ x \in \Re^n \;\middle|\; g_j(x) \le 0,\ j = 1, 2, \cdots, p \right\} \tag{456}$$

is called the feasible set. An element x in S is called a feasible solution.

Definition A.2： A feasible solution x^* is called the optimal solution of SOP if and only if

$$f(x^*) \geq f(x) \tag{A-1}$$

for any feasible solution x.

One of the outstanding contributions to mathematical programming was known as the Kuhn-Tucker conditionsA-2. In order to introduce them, let us give some definitions. An inequality constraint $g_j(x) \leq 0$ is said to be active at a point x^* if $g_j(x^*) = 0$. A point x^* satisfying $g_j(x^*) \leq 0$ is said to be regular if the gradient vectors $\nabla g_j(x)$ of all active constraints are linearly independent.

Let x^* be a regular point of the constraints of SOP and assume that all the functions $f(x)$ and $g_j(x), j = 1, 2, \cdots, p$ are differentiable. If x^* is a local optimal solution, then there exist Lagrange multipliers $\lambda_j, j = 1, 2, \cdots, p$ such that the following Kuhn-Tucker conditions hold,

$$\begin{cases} \nabla f(x^*) - \sum\limits_{j=1}^{p} \lambda_j \nabla g_j(x^*) = 0 \\ \lambda_j g_j(x^*) = 0, \quad j = 1, 2, \cdots, p \\ \lambda_j \geq 0, \quad j = 1, 2, \cdots, p. \end{cases} \tag{A-2}$$

If all the functions $f(x)$ and $g_j(x), j = 1, 2, \cdots, p$ are convex and differentiable, and the point x^* satisfies the Kuhn-Tucker conditions (A-2), then it has been proved that the point x^* is a global optimal solution of SOP.

A.1.1 Linear Programming

If the functions $f(x), g_j(x), j = 1, 2, \cdots, p$ are all linear, then SOP is called a linear programming.

The feasible set of linear is always convex. A point x is called an extreme point of convex set S if $x \in S$ and x cannot be expressed as a convex combination of two points in S. It has been shown that the optimal solution to linear programming corresponds to an extreme point of its feasible set provided that the feasible set S is bounded. This fact is the basis of the simplex algorithm which was developed by Dantzig as a very efficient method for solving linear programming.

Table 1 This is an example for manually numbered table, which would not appear in the list of tables

Network Topology		# of nodes	# of clients			Server
GT-ITM	Waxman Transit-Stub	600	2%	10%	50%	Max. Connectivity
Inet-2.1		6000				
Xue	Rui	Ni	ThuThesis			
	ABCDEF					

Roughly speaking, the simplex algorithm examines only the extreme points of the feasible set, rather than all feasible points. At first, the simplex algorithm selects an extreme point as the initial point. The successive extreme point is selected so as to improve the objective function value. The procedure is repeated until no improvement in objective function value can be made. The last extreme point is the optimal solution.

A.1.2 Nonlinear Programming

If at least one of the functions $f(x), g_j(x), j = 1, 2, \cdots, p$ is nonlinear, then SOP is called a nonlinear programming.

A large number of classical optimization methods have been developed to treat special-structural nonlinear programming based on the mathematical theory concerned with analyzing the structure of problems.

Figure 1 This is an example for manually numbered figure, which would not appear in the list of figures

Now we consider a nonlinear programming which is confronted solely with maximizing a real-valued function with domain $\Re^n$. Whether derivatives are available or not, the usual strategy is first to select a point in $\Re^n$ which is thought to be the most likely place where the maximum exists. If there is no information available on which to base such a selection, a point is chosen at random. From this first point an attempt is made to construct a sequence of points, each of which yields an improved objec-

tive function value over its predecessor. The next point to be added to the sequence is chosen by analyzing the behavior of the function at the previous points. This construction continues until some termination criterion is met. Methods based upon this strategy are called ascent methods, which can be classified as direct methods, gradient methods, and Hessian methods according to the information about the behavior of objective function f. Direct methods require only that the function can be evaluated at each point. Gradient methods require the evaluation of first derivatives of f. Hessian methods require the evaluation of second derivatives. In fact, there is no superior method for all problems. The efficiency of a method is very much dependent upon the objective function.

A.1.3 Integer Programming

Integer programming is a special mathematical programming in which all of the variables are assumed to be only integer values. When there are not only integer variables but also conventional continuous variables, we call it mixed integer programming. If all the variables are assumed either 0 or 1, then the problem is termed a zero-one programming. Although integer programming can be solved by an exhaustive enumeration theoretically, it is impractical to solve realistically sized integer programming problems. The most successful algorithm so far found to solve integer programming is called the branch-and-bound enumeration developed by Balas (1965) and Dakin (1965). The other technique to integer programming is the cutting plane method developed by Gomory (1959).

Uncertain Programming (*BaoDing Liu, 2006.2*)

References

NOTE: these references are only for demonstration, they are not real citations in the original text.

[1] Donald E. Knuth. The TEXbook. Addison-Wesley, 1984. ISBN: 0-201-13448-9

[2] Paul W. Abrahams, Karl Berry and Kathryn A. Hargreaves. TEX for the Impatient. Addison-Wesley, 1990. ISBN: 0-201-51375-7

[3] David Salomon. The advanced TEXbook. New York : Springer, 1995. ISBN:0-387-94556-3

附录 B　外文资料的调研阅读报告或书面翻译

B.1　单目标规划

北冥有鱼，其名为鲲。鲲之大，不知其几千里也。化而为鸟，其名为鹏。鹏之背，不知其几千里也。怒而飞，其翼若垂天之云。是鸟也，海运则将徙于南冥。南冥者，天池也。

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}, y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \tag{123}$$

吾生也有涯，而知也无涯。以有涯随无涯，殆已！已而为知者，殆而已矣！为善无近名，为恶无近刑，缘督以为经，可以保身，可以全生，可以养亲，可以尽年。

B.1.1　线性规划

庖丁为文惠君解牛，手之所触，肩之所倚，足之所履，膝之所倚，砉然响然，奏刀騞然，莫不中音，合于桑林之舞，乃中经首之会。

表 1　这是手动编号但不出现在索引中的一个表格例子

Network Topology		# of nodes	# of clients			Server
GT-ITM	Waxman Transit-Stub	600	2%	10%	50%	Max. Connectivity
Inet-2.1		6000				
Xue	Rui	Ni	ThuThesis			
	ABCDEF					

文惠君曰：“嘻，善哉！技盖至此乎？”庖丁释刀对曰：“臣之所好者道也，进乎技矣。始臣之解牛之时，所见无非全牛者；三年之后，未尝见全牛也；方今之时，臣以神遇而不以目视，官知止而神欲行。依乎天理，批大郤，导大窾，因其固然。技经肯綮之未尝，而况大軱乎！良庖岁更刀，割也；族庖月更刀，折也；今臣之刀十九年矣，所解数千牛矣，而刀刃若新发于硎。彼节者有间而刀刃者无厚，以无厚入有间，恢恢乎其于游刃必有余地矣。是以十九年而刀刃若新发于硎。虽然，每至于族，吾见其难为，怵然为戒，视为止，行为迟，动

刀甚微，謋然已解，如土委地。提刀而立，为之而四顾，为之踌躇满志，善刀而藏之。”

文惠君曰：“善哉！吾闻庖丁之言，得养生焉。”

B.1.2 非线性规划

孔子与柳下季为友，柳下季之弟名曰盗跖。盗跖从卒九千人，横行天下，侵暴诸侯。穴室枢户，驱人牛马，取人妇女。贪得忘亲，不顾父母兄弟，不祭先祖。所过之邑，大国守城，小国入保，万民苦之。孔子谓柳下季曰：“夫为人父者，必能诏其子；为人兄者，必能教其弟。若父不能诏其子，兄不能教其弟，则无贵父子兄弟之亲矣。今先生，世之才士也，弟为盗跖，为天下害，而弗能教也，丘窃为先生羞之。丘请为先生往说之。”

图 1 这是手动编号但不出现索引中的图片的例子

柳下季曰：“先生言为人父者必能诏其子，为人兄者必能教其弟，若子不听父之诏，弟不受兄之教，虽今先生之辩，将奈之何哉？且跖之为人也，心如涌泉，意如飘风，强足以距敌，辩足以饰非。顺其心则喜，逆其心则怒，易辱人以言。先生必无往。”

孔子不听，颜回为驭，子贡为右，往见盗跖。

B.1.3 整数规划

盗跖乃方休卒徒大山之阳，脍人肝而餔之。孔子下车而前，见谒者曰：“鲁人孔丘，闻将军高义，敬再拜谒者。”谒者入通。盗跖闻之大怒，目如明星，发上指冠，曰：“此夫鲁国之巧伪人孔丘非邪？为我告之：尔作言造语，妄称文、武，冠枝木之冠，带死牛之胁，多辞缪说，不耕而食，不织而衣，摇唇鼓舌，擅生是非，以迷天下之主，使天下学士不反其本，妄作孝弟，而侥幸于封侯富贵者也。子之罪大极重，疾走归！不然，我将以子肝益昼餔之膳。”

附录 C　其它附录

前面两个附录主要是给本科生做例子。其它附录的内容可以放到这里，当然如果你愿意，可以把这部分也放到独立的文件中，然后将其 \input 到主文件中。

在学期间参加课题的研究成果

个人简历

1990 年 4 月 25 日出生于北京市。

2008 年 9 月考入清华大学计算机系，2012 年 7 月本科毕业并获得工学学士学位。

2012 年 9 月免试进入清华大学计算机系攻读硕士学位至今。

发表的学术论文

[1] 一种灵活的IaaS云服务租户带宽保障模型，计算机工程与应用（已录用）