应用感知的数据中心网络多租户共享方法

李 妍 郭得科 曹晓丰 陈洪辉

(国防科技大学信息系统工程重点实验室 长沙 410073)

摘 要 在多租户数据中心,来自不同租户的应用程序共享并竞争使用网络资源.网络共享策略会对应用程序端到端的性能(如作业完成时间、吞吐量等)产生直接的影响.为了衡量租户应用程序的整体数据传输速率,本文引入进度(Progress)的指标.该指标被定义为租户应用程序在所有链路上经需求标准化后的最小带宽分配量,反映的是租户能够完成其数据传输的最慢速率.通过最大程度地提高租户进度,可以优化上层应用程序的执行时间等性能.先前的大多数工作都集中在实现网络共享的公平性、可预测性和效率之间的权衡,忽略了提高租户的长期进度.本文观察发现应用程序放置于租户所租赁的不同虚拟机上会形成不同的带宽需求分布,进而影响后续带宽分配所能够获得的最优进度.通过理论分析我们证明了获得所有租户最优进度的关键在于最小化网络瓶颈链路上的带宽需求.基于此,本文提出应用感知的网络多租户共享方法,通过联合优化任务放置和带宽分配的过程,该方法最大化所有租户的进度,并在优势资源公平性限制下最大化网络利用率.实验结果证明,与目前的最新带宽分配方法相比,本文将租户整体进度提高了 85.6%~107.7%,网络链路利用率提高了 71.2%~112.4%.

关键词 网络共享;带宽分配;多租户;应用感知;任务放置;进度

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2021.01363

Application-Aware Network Sharing for Multi-Tenant Data Center

LI Yan GUO De-Ke CAO Xiao-Feng CHEN Hong-Hui

(*Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha* 410073)

Abstract In the multi-tenant shared cloud, tenants usually deploy their applications on virtual machines (VMs) rent from the cloud. Such applications generate varying amounts of traffic between their VM pairs. Applications of different tenants compete for the shared network, resulting in significant unpredictable performance which not only makes tenant costs incalculable but also causes revenue loss for cloud service providers. Sharing network among tenants is challenging because the bandwidth obtained by tenants depends on a variety of complex factors, such as system load, VM placement, and the oversubscription ratio of the data center network. There is a broad consensus is that network sharing mechanisms should be fair, predictable, and efficient. Specifically, tenants expect to guarantee the minimum bandwidth to achieve performance predictability, while cloud providers pursue high network utilization and strategy-proofness. Research shows that there are strong trade-offs among these three goals. Most previous work has focused on achieving tradeoffs while ignoring the long-term goal of improving application performance. In fact, network allocation decisions have a significant impact on the end-to-end performance (e.g., job completion time, throughput, etc.) of applications. To measure the overall data transfer rate of a tenant, we introduce the metric *progress*, which is defined as the

收稿日期:2019-11-20;在线发布日期:2020-06-25.本课题得到国家重点研发计划(2018YFE0207600)、国家自然科学基金联合基金重点项目(U19B2024)资助.**李 妍**,博士研究生,主要研究方向为数据中心网络、云计算.E-mail:liyan10@nudt.edu.cn.**郭得科**(通信作者),博士,教授,博士生导师,中国计算机学会(CCF)杰出会员,主要研究领域为网络计算与系统、分布式计算与系统、移动计算.E-mail:guodeke@gmail.com.**曹晓丰**,博士研究生,主要研究方向为边缘计算、大数据分析处理.**陈洪辉**,博士,教授,博士生导师,主要研究领域为体系结构、云计算、信息检索.

minimum demand-normalized bandwidth allocation on all network links. The progress is an essential metric to indicate how fast a tenant can complete data transfer. By maximizing the tenant's progress, application performance such as execution time can be optimized. Currently, much work has proved that given the bandwidth demand vectors of all tenants, the state-of-the-art bandwidth allocation schemes i.e., DRF and HUG, can obtain the optimal progress. However, we found that by changing the bandwidth demand distribution, the tenant's progress and isolation guarantee can be further improved. The insight is to tune and optimize the placement of tenants' applications to cause the desired traffic pattern by using information about the underlying network and traffic patterns of tenants' applications. Based on the above observations, we propose a novel application-aware network sharing mechanism. By jointly optimizing the task placement and bandwidth allocation, we maximize the progress of all tenants and improve network utilization under the constraint of dominant resource fairness. Specifically, this application-aware network sharing mechanism includes two stages: (1) task placement on different VMs results in different distributions of bandwidth demand, which in turn determines the optimal progress that can be obtained by subsequent bandwidth allocation. Through theoretical analysis, we prove that the key to obtaining the optimal progress is to minimize the bandwidth demands on the bottleneck link; and (2) bandwidth allocation determines how to allocate bandwidth among tasks of different tenants to obtain the optimal isolation guarantee and maximize network utilization. This optimization problem is NP-hard integer programming. Through problem conversion, we devise a task placement algorithm for finding a local optimal solution and a bandwidth allocation algorithm to achieve optimal progress. We conduct extensive evaluations spanning various environment settings and application dynamics. The result demonstrates that our method can outperform the state-of-the-art scheme by 85.6%—107.7% higher tenant progress and 71.2%—112.4% higher network utilization.

Keywords network sharing; bandwidth allocation; multi-tenant; application-aware; task placement; progress

1 引 言

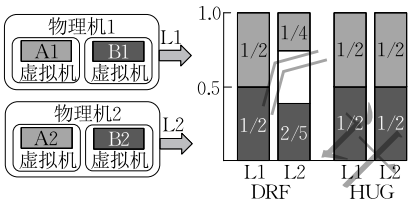
在基础架构即服务(IaaS,即"云")数据中心内,租户(Tenant)按固定费率租赁虚拟机(Virtual Machine,VM)并在其上部署自己的应用程序.在这种模式下,租户的计算和存储资源与云中的其他租户之间是隔离使用的;相反网络资源却以"尽力服务"模式由租户之间共享[1-2].因此,租户可获得的网络带宽受限于一系列复杂的因素,如网络负载、虚拟机的放置以及数据中心网络的超额预定比率等.这造成了租户间网络性能的显著差异,从而严重损害应用程序的性能,不仅使租户成本变得不可预测,同时也造成云服务提供商的收入损失.

近年来,针对多租户数据中心内部网络的共享研究已成为数据中心资源管理和分配的重要领域,其中一个广泛共识是租户之间的网络共享方案应该公平、可预测且高效.具体地,租户期望保证其应用程序的最低带宽,也就是隔离保证(Isolation Guarantee),以实现性能可预测性,而云提供商则追求网络高利用率和策略安全性.研究表明,这三种共享目标之间存在很强的权衡取舍[3].大多数先前工作都集中在实现公平性、可预测性和效率之间的权衡,却忽略了提高应用程序性能的长期目标.

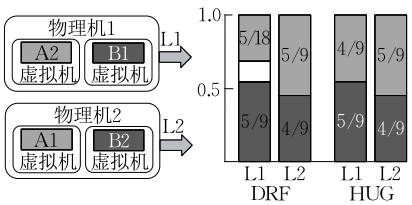
事实上网络共享决策会对应用程序端到端的性能(如作业完成时间、吞吐量等)产生直接的影响.这里,我们将应用程序看作是一组任务的集合,任务之间需要进行数据传输,任务可获得的带宽在很大程度上取决于同一虚拟机上运行的其他任务以及同一物理机上的其他虚拟机.为了衡量租户的整体数据传输率,我们引入进度(Progress)指标.根据HUG[4]中的定义,进度是租户应用程序在所有网络链路上经需求标准化后的最小带宽分配量,其值取决于租户的带宽需求以及分配到的带宽.

通过最大化租户的进度,可以优化上层应用程序执行时间等性能.具体地,数据中心网络涉及多条

链路，且租户任务间的带宽需求在这些链路上是弹性且相关的. 如图 1(a)所示，租户 A 和 B 的应用程序各拥有两个任务，(A1，A2)和(B1，B2)，其中 A1 和 B1 位于物理机 1，分别需要 1 Gbps 和 1 Gbps 的上行带宽①，A2 和 B2 位于物理机 2，分别需要 0.5 Gbps 和 0.8 Gbps 的上行带宽. 租户任务在不同链路上的带宽需求以向量的形式表示，则租户 A 的需求向量为(1，0.5)，租户 B 为(1，0.8). 假设物理机的上行带宽容量均为 1 Gbps，则根据当前最先进的带宽分配方法 DRF[3] 或 HUG[4] 计算可得，租户 A 的进度为 $M_A=\min(0.5/1,0.25/0.5)=1/2$，且 $M_A=M_B=1/2$. 同时我们定义隔离保证为所有租户中最低的进度，即 $\min(M_A,M_B)=1/2$.



(a) 租户A和B的进度均为1/2



(b) 租户A和B的进度均为5/9

图 1　租户带宽需求及分配示例，在不同任务放置下DRF和HUG 的带宽分配不同，获得的租户进度也不同

目前大量的工作[3-4] 已证明在给定所有租户的带宽需求向量后，DRF 或 HUG 的带宽分配均可获得最佳隔离保证. 但是我们发现，通过更改租户的带宽需求向量，可以进一步提高租户的进度和隔离保证. 例如，若我们将租户 A 的任务 A1 和 A2 的位置互换，如图 1(b)所示，则任务 A1 在物理机 1 上需要 0.5 Gbps 的上行带宽，任务 A2 在物理机 2 上需要 1 Gbps. 这种情况下，根据 DRF 或 HUG 计算可得到 $M_A=M_B=5/9$，大于图 1(a)的 1/2. 这使得我们意识到，任务放置位置的不同会影响租户所获得的进度. 通过进一步地理论分析，我们证明获得所有租户最优进度的关键在于，寻找一种任务放置方案使得网络各链路的带宽需求尽可能地“平均”，更公式化地，即最小化网络瓶颈链路上的带宽需求.

基于以上观察，本文中我们提出一种新颖的基于应用感知的网络共享方法，通过联合优化任务放置和带宽分配的过程，我们最大化所有租户的进度，并在优势资源公平性限制下尽可能提高网络利用率. 具体地，本文的网络分配机制包括两个相互依赖的阶段：任务放置于其所租赁的不同虚拟机上会形成不同的带宽需求分布，进而决定了后续带宽分配所能够获得的最优进度；而带宽分配策略决定如何在不同租户的任务之间分配带宽以获得最优进度，同时尽可能最大化网络利用率. 该优化问题是 NP 难的整数规划，通过问题转换，本文提出了一种用于找到局部最优解的任务放置算法以及达到最优进度的带宽分配算法. 文章的最后，我们通过大规模仿真实验验证了我们方法的有效性，与现有网络共享方法 DRF 和 HUG 相比，本文提出的基于应用感知的数据中心网络多租户共享方法将租户的整体进度提高了 85.6%～107.7%，并将网络利用率提高了 71.2%～112.4%.

本文工作的贡献包含以下几点：(1) 提出一种基于应用感知的网络多租户共享方法. 不同于已有的方法，我们通过调节应用程序任务的放置，在优势资源公平性限制下获得最优的租户进度和网络利用率；(2) 在对任务进行放置的过程中，我们的策略是最小化瓶颈链路的带宽需求，并证明这种放置可以确保后续的带宽分配获得最优的租户进度；(3) 通过考虑网络的动态性，我们提出的方法可以确保获得长期的高进度、高网络利用率以及应用程序性能.

本文第 2 节介绍相关工作与方法；第 3 节给出我们的研究动机并举例说明；第 4 节给出问题的详细描述，分别对任务放置和带宽分配建模并给出模型求解分析；第 5 节给出应用感知的网络多租户共享的算法详细描述；第 6 节通过大规模仿真实验验证本文方法的性能；最后第 7 节为全文总结.

2　相关工作

面向云的网络共享研究大致可分为三类：(1) 数据流层面的共享，它支持单个 TCP 连接或 TCP 端口层面的带宽和/或延迟保证；(2) 虚拟机层面的共享，它着重于确保不同租户虚拟机之间的带宽分配；(3) 租户层面的共享，它聚焦租户的所有虚拟机的聚合带宽的共享. 以上三个层面的网络共享粒度逐渐降低，租户表达带宽需求的难度也逐渐降低.

① 任务上行带宽为“该任务每秒钟传送给其他任务的最大数据量”.

在第一类研究中,TCP 拥塞控制和每流公平共享算法(Per-flow Fairness)是被广泛使用的两种网络分配方法. TCP 拥塞控制限制发送端传输数据流的速率,以避免网络的过载. 每流公平共享算法则将网络带宽在所有数据流之间公平分配. 这种传统的分配策略以隔离的方式决定每一个数据流的带宽分配量,并简单地认为数据流获得带宽越多,性能越好.

在第二类研究中,Shieh 等人[5]提出的 Seawall 是一种基于虚拟机管理程序(hypervisor)的机制,该机制通过控制网络边缘每台服务器的虚拟化层中的速率限制器,在数据中心网络的每条链路上分配带宽. Seawall 根据每台虚拟机的权重分配拥塞链路上的带宽,从而实现租户虚拟机的最大最小公平性. Guo 等人[6]在 SecondNet 中提出"虚拟数据中心(Virtual Data Center,VDC)"的抽象模型,并采取静态保留的网络共享机制为虚拟机与虚拟机之间的流量提供带宽保证. Ballani 等人[7]提出的 Oktopus 扩展了 VDC 模型,提出"虚拟超额预定集群(Virtual Oversubscribed Cluster,VOC)"模型,它结合基于管理程序的速率限制和虚拟机放置策略,从而满足请求的带宽需求(或拒绝无法满足的请求). Rodrigues 等人[8]在 Gatekeeper 中提出了一种基于虚拟机的软管模型,即假设租户的所有虚拟机都由无阻塞交换机连接,并为每个虚拟机分别分配入口和出口带宽. Gatekeeper 使用基于虚拟机管理程序的速率限制和基于反馈的机制来防止远程虚拟机向其他虚拟机发送超过允许的流量.

在第三类研究中,Lam 等人[9]提出的 NetShare 为租户分配不同的权重(如根据价格),并按照权重分配拥塞链路的带宽. 由于租户的权重在整个网络中都维持不变,但是拥塞发生在链路层面,并且可能涉及到不同数量的虚拟机,因此 NetShare 不能提供直接的带宽保证. Popa 等人[10]在 FairCloud 中详细分析了云网络共享设计空间中的关键要求和属性,以及这些要求之间的固有权衡. FairCloud 认为,除了最低带宽保证之外,云网络还应在需求未满足的情况下提供网络高利用率以及网络比例,即租户之间的带宽分配应与租户虚拟机数量成正比. 在此基础之上作者提出 PS-L、PS-P 和 PS-N 三种分配策略实现不同的权衡,其中 PS-P 算法在树型结构网络上实现租户层面网络共享的一系列最优属性. Raghavan 等人[3]提出了 DRL,它对给定租户的所有站点所产生的流量总和实行限制. DRL 将最大最小公平性创造性地应用在多资源分配上,其核心思想是使所有用户的优势资源份额均等,其中优势资源指的是所有资源类型中用户所需百分比份额最大的资源. DRL 能够实现最佳的隔离保证,但是由于优势资源公平性的限制,DRL 具有较低的网络利用率,在一些极端情况下 DRL 的利用率甚至趋于 0.

与我们工作最相似的是 HUG[4],HUG 将最大最小公平性扩展到具有相关需求且弹性的多链路网络共享中. HUG 直接采用了 DRL 的优势资源公平性来最大程度地提高隔离保证,但通过考虑弹性带宽需求,HUG 尽可能地提高网络利用率和应用程序性能. 但是,我们发现 DRL 和 HUG 获得的最佳隔离保证依赖于当前网络内各租户的应用程序带宽需求向量,而通过改变任务的放置,我们可以改变并求得最优的带宽需求向量,从而进一步提升隔离保证、网络利用率和应用程序性能.

值得注意的是,上述三类研究在不同粒度上考虑带宽资源的共享,其中数据流和虚拟机层面的共享均不能确保应用程序的性能和公平性. 一方面,对于数据流层面的共享,假设两个租户共享同一条网络链路,租户 A 的应用程序采用多对多的通信方式(如产生 99 个 TCP 连接),租户 B 的应用程序采用一对一的通信方式,数据流层面的共享尝试在流之间平均分配带宽,即将 99% 的带宽分配给租户 A,1% 的带宽分配给租户 B,这种分配方式对租户显然是不公平的;另一方面,在虚拟机层面的共享中租户所获得的带宽取决于其虚拟机数量、位置以及其他租户的流量分布等,假设租户 A 和 B 在物理机上分别拥有一个虚拟机,租户 A 的虚拟机接收来自多个其他虚拟机的数据,租户 B 的虚拟机只接收来自 1 个虚拟机的数据,根据虚拟机层面的共享策略(如 Seawall),租户 A 将会获得该物理机上绝大多数的带宽资源,因此该层面的网络共享同样不能保证上层应用程序获得的带宽. 鉴于以上原因,本文从租户的角度定义网络共享的目标及理想机制.

3 研究动机

近年来,针对多租户的数据中心内部网络共享研究已成为数据中心资源管理和分配的重要领域,其中一个广泛共识是租户之间的网络共享解决方案应该公平、可预测且高效. 根据先前工作[10-12]所定义,云"租户"间共享网络的理想解决方案应满足以下三个要求:一是隔离保证,指不管其他租户的网络利用率如何,都应为租户提供对于他们购买的虚拟

机可以期望的最小网络带宽的保证；二是高利用率，旨在存在未满足的需求时最大化网络利用率；三是网络比例，指网络资源应按租户的付款比例进行分配. 先前的大多数网络共享工作都集中在实现公平性、可预测性和效率之间的权衡，却忽略了提高应用程序性能这种长期目标.

事实上，网络共享决策会对应用程序端到端的性能（如作业完成时间、吞吐量等）产生直接且重要的影响. 实际上数据中心网络涉及许多条链路，为简单起见这里我们仅关心访问链接的带宽，即物理机的上行带宽和下行带宽. 租户在不同链路上的需求是弹性且相关的，其中弹性是指一方面租户可以在网络可用容量低于其带宽需求的链路上进行数据传输，另一方面若仍有空闲带宽，租户可以使用超过其需求量的带宽，以加速数据传输过程；相关性是指对于租户在链路 i 上发送的每一比特数据，该租户在链路 j 上发送至少 α 比特.

为了衡量租户应用程序的整体数据传输率，我们引入进度（Progress）的指标，该指标反映的是租户完成数据传输的速度. 结合 HUG[4] 中的定义，租户的进度是其在所有网络链路上获得的经需求标准化后最小的带宽分配量. 具体地，记租户 K 在链路 i 上的带宽需求为 d_K^i，获得的带宽分配量为 r_K^i，则该租户进度 M_K 为

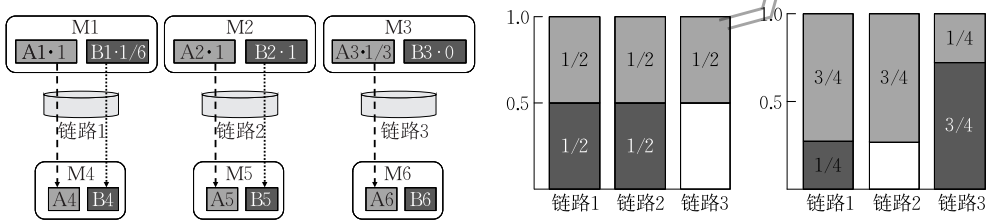
$$M_K=\min_{i\in I}\left\{\frac{r_K^i}{d_K^i}\right\} \tag{1}$$

其中，I 为全网络链路的集合，租户在链路上的带宽需求为其所有任务在该条链路上的带宽需求总和. 根据定义可知，租户的进度由其在所有网络链路上的需求和在相应链路上得到的带宽决定.

进一步地，由上述数学公式可知，进度可以理解为瓶颈链路上分配给租户的带宽量，而这决定了数据传输的最慢完成时间. 因此，通过最大程度地提高租户进度，我们能够提高上层应用程序的性能. 为探讨租户进度提高的可能方案，我们给出图 2 的示例. 该网络中包含两个租户 A 和 B，每个租户的应用程序都包含有 6 个任务，并且各自租赁六个位于不同物理计算机上的虚拟机. 同一租户的虚拟机之间相互通信，并与其他租户的虚拟机竞争使用物理机的上行和下行带宽资源. 为简化示例讨论，这里我们假设虚拟机上只放置唯一的任务. 租户的任务在虚拟机中的放置情况如图 2(a)所示. 这里我们仅考虑上层物理机的出口带宽（即上行带宽）的分配情况，值得注意的是任务的出口带宽需求对下层任务来说相当于其入口带宽（下行带宽）需求. 因此租户 A 的带宽需求向量为(1,1,1/3)，租户 B 的带宽需求向量则为(1/6,1,0)，这里我们采用标准化的带宽需求，即实际需求与链路容量的比值. 根据 HUG 的分配策略可计算得到，租户 A 得到的带宽分配为(1/2,1/2,1/2)，租户 B 得到的带宽为(1/2,1/2,0). 根据进度的定义，租户 A 的进度为

$$M_A=\min\left(\frac{1/2}{1},\frac{1/2}{1},\frac{1/2}{1/3}\right)=1/2,$$

同样计算可得到租户 B 的进度也为 1/2.



(a) 来自租户A和B的不同虚拟机间的带宽需求　(b) HUG带宽分配　(c) 基于应用感知的带宽分配

图 2　研究动机示例（任务的放置决定租户所能获得的最佳进度）

目前已有大量的工作可以证明在给定租户的带宽需求向量后，DRF 或 HUG 的带宽分配可获得最佳隔离保证[3-4]. 但我们发现，若改变任务的放置位置，例如将任务 B2 和 B3 位置互换，则租户 B 的需求向量变为(1/6,0,1). 重新根据 HUG 的分配策略计算可得租户 A 的带宽分配为(3/4,3/4,1/4)，租户 B 为(1/4,0,3/4). 进一步根据式(1)计算可得租户 A 和 B 的进度为 3/4. 也就是说，通过更改任务的放置，我们将租户进度由 1/2 提高至 3/4，与此同时，网络的利用率由 83.3%提高至 91.7%.

基于以上观察，本文发现通过更改任务位置，可以进一步改善进度. 由于虚拟机的放置是一个复杂的问题，涉及到一系列的约束，例如数据局部性约束、容错性约束等. 因此本文仅考虑任务的放置. 具体地，本文提出一种基于应用感知的网络多租户共享方法，通过联合优化任务放置和带宽分配的过程，最大化所有租户的进度，并在优势资源公平性限制下尽可能提高网络利用率. 这里的应用感知旨在强

调两层含义:一是在感知不同应用程序带宽需求的基础上,通过调节应用程序任务的放置,我们可以进一步优化网络带宽分配;二是区别于原有网络共享方法旨在实现公平性、可预测性和效率之间的权衡,我们强调网络共享方法需要重点考虑提高应用程序性能这类长期目标(如进度).

4 任务放置和带宽分配的建模

4.1 问题描述

本文中我们考虑基础架构即服务(IaaS,即"云")数据中心,在这类数据中心内租户为每台虚拟机按租赁时间支付固定费率[13-14].为了简化说明,我们在本文中假设所有虚拟机(在硬件资源方面)都是相同的,并且价格相同.根据生产数据中心收集的统计数据[15],核心网络很少会出现严重或持续的拥塞,而网络边缘通常会被完全占用.鉴于此类观察,为简单起见,我们假设拥塞仅发生在网络边缘.因此,我们关心的网络资源仅限于访问链接的带宽[4,15],特定的网络拓扑[16-17]或路由协议[18]对网络共享问题没有影响.基本符号定义于表 1.

表 1 基本符号定义

符号	意义说明
n	云环境中物理计算机的总数量
m	云环境中租户的总数量
L_K	租户 K 拥有的虚拟机集合
S_K	租户 K 需要运行的任务集合
$(c_{K,l,p}^{\text{cpu}},c_{K,l,p}^{\text{mem}})$	虚拟机的计算及存储资源容量
$c_p^{\text{netIn}},c_p^{\text{netOut}}$	物理计算机 p 的下行和上行带宽容量
$(d_{i,K}^{\text{cpu}},d_{i,K}^{\text{mem}})$	租户 K 的任务 i 的计算及存储资源需求量
$\boldsymbol{d}_K^{\text{net}}=[d_K^{i,j}]$	租户 K 的网络资源需求矩阵
M_K	租户 K 的进度
$d_{i,K}^{\text{netOut}},d_{i,K}^{\text{netIn}}$	租户 K 的任务 i 的下行及上行带宽需求
$X_{i,K}^{l,p}$	0-1 变量,指示租户 K 任务 i 是否放置虚拟机 l
$R_{K,p}^{\text{netOut}},R_{K,p}^{\text{netIn}}$	租户 K 在物理机 p 上所分配的保证带宽
$G_{K,p}^{\text{netOut}},G_{K,p}^{\text{netIn}}$	租户 K 在物理机 p 上所分配的实际带宽

基于以上观察,我们将数据中心网络抽象为多台物理机通过无阻塞交换机相连接.一台物理机可以托管一台或多台虚拟机,这些虚拟机可能属于不同的租户.物理机具有多种类型的资源,这里我们考虑两类资源:(1)计算及存储资源,如 CPU、内存、磁盘等,这类资源在虚拟机之间是隔离使用的,因此我们为每台虚拟机定义一个配置向量$(c_{K,l,p}^{\text{cpu}},c_{K,l,p}^{\text{mem}})$,该向量指定了其拥有的计算及存储资源容量;(2)网络资源,如上行及下行带宽,这类资源由所有虚拟机之间共享,因此我们定义 $c_p^{\text{netIn}},c_p^{\text{netOut}}$ 分别为物理机 p 的下行和上行带宽容量.

通常租户会租用一组虚拟机来运行其应用程序.这里,我们将应用程序视为一组任务的集合.只要仍然满足资源约束,每台虚拟机上可以部署多个任务,任务之间存在数据传输.假设租户 K 租赁 L_K 台虚拟机来运行其 S_K 个任务,虚拟机运行在不同的物理机上.记租户 K 的任务 i 的计算及存储资源需求分别为$(d_{i,K}^{\text{cpu}},d_{i,K}^{\text{mem}})$,与此同时,记租户 K 的网络资源需求为 $S_K\times S_K$ 的矩阵 $\boldsymbol{d}_K^{\text{net}}=[d_K^{i,j}]$,其中 $d_K^{i,j}$ 代表的是租户 K 的任务 i 到任务 j 的带宽需求.

在网络资源分配中,任务可获得的带宽在很大程度上取决于在同一虚拟机上运行的其他任务以及同一物理机上的其他虚拟机.不失一般性地,本文假设同一应用的不同任务放置在不同的物理机上.基于应用感知的网络共享方法应包含两方面的内容:一是合理地放置任务,以实现最优的租户进度;二是合理地分配带宽,以实现其他网络共享目标.更形式化地,上述问题可以定义为:

定义 1. 基于应用感知的多租户网络共享方法.给定 m 个租户的资源需求 $\boldsymbol{D}=\{\boldsymbol{D}_1,\cdots,\boldsymbol{D}_m\}$,其所有虚拟机的计算及存储资源容量 $\boldsymbol{C}=\{\boldsymbol{C}_{L_1},\cdots,\boldsymbol{C}_{L_m}\}$以及 n 台物理机网络资源容量 $\boldsymbol{C}^{\text{net}}=\{\boldsymbol{C}_1^{\text{net}},\cdots,\boldsymbol{C}_n^{\text{net}}\}$,基于应用感知的网络共享策略 $\boldsymbol{P}$ 是将 m 个租户的任务集合 $\boldsymbol{S}=\{\boldsymbol{S}_1,\cdots,\boldsymbol{S}_m\}$ 按照 $\boldsymbol{X}=\{\boldsymbol{X}_1,\cdots,\boldsymbol{X}_m\}$ 放置在其租赁的相应虚拟机 $\boldsymbol{L}=\{\boldsymbol{L}_1,\cdots,\boldsymbol{L}_m\}$ 上,同时为 m 个租户分配带宽 $\boldsymbol{R}=\{\boldsymbol{R}_1,\cdots,\boldsymbol{R}_m\}$,即

$$\boldsymbol{P}(\boldsymbol{C},\boldsymbol{C}^{\text{net}},\boldsymbol{D})=\{\boldsymbol{X},\boldsymbol{R}\} \tag{2}$$

其中,$S_K\times L_K$ 的矩阵 $\boldsymbol{X}_K=[X_K^{i,I}]$ 代表的是租户 K 的任务 i 的放置.

4.2 任务放置模型

为了将任务放置问题与先前的理论工作联系起来,我们将其转换为通用的优化框架.由第 3 节中讨论可知,任务放置决定了后续带宽分配策略所能够获得的最优租户进度.因此,任务放置问题可抽象为在满足资源等约束的前提下,如何将租户的任务放置在不同的虚拟机上从而实现全网络的最优隔离保证.该问题可建模为多维资源分配问题.

首先,我们考虑资源容量与需求模型.关于网络资源,由于数据中心网络涉及多台物理计算机的上行链路和下行链路,因此任务间的带宽需求通常是相关的且是弹性的.我们定义 $d_{i,K}^{\text{netOut}}$ 和 $d_{i,K}^{\text{netIn}}$ 分别为租

户 K 的任务 i 的上行和下行带宽需求，即

$$d_{i,K}^{\text{netOut}}=\sum_j d_K^{i,j} \tag{3}$$

$$d_{i,K}^{\text{netIn}}=\sum_j d_K^{j,i} \tag{4}$$

具体地，对租户 K 的任一任务 i 赋予其需求向量 $\boldsymbol{d}_{i,K}=(d_{i,K}^{\text{cpu}},d_{i,K}^{\text{mem}},d_{i,K}^{\text{netOut}},d_{i,K}^{\text{netIn}})$，分别表示该任务所需求的计算资源、存储资源、上行带宽和下行带宽；对虚拟机 l，定义容量向量为 $\boldsymbol{c}_{K,l,p}=(c_{K,l,p}^{\text{cpu}},c_{K,l,p}^{\text{mem}})$，分别表示该虚拟机的计算资源和存储资源容量，其中下标 p 表明该虚拟机所位于的物理服务器；对物理机 p，其网络资源容量向量为 $\boldsymbol{c}_p=(c_p^{\text{netOut}},c_p^{\text{netIn}})$，分别表示该物理机的上行带宽和下行带宽容量.

其次，为了对任务放置和虚拟机的选择进行建模，我们构造放置变量 $\boldsymbol{X}_K=[X_{i,K}^{l,p}]$，以指示租户 K 的任务 i 如何放置在虚拟机上，具体可表示为

$$\begin{cases}X_{i,K}^{l,p}=1, & \text{若租户 } K \text{ 的任务 } i \text{ 放在虚拟机 } l\\ X_{i,K}^{l,p}=0, & \text{否则}\end{cases} \tag{5}$$

其中，$\boldsymbol{X}_K=[X_{i,K}^{l,p}]$ 为 $S_K\times L_K$ 的矩阵，$l\in L_K$ 表示租户 K 的任务只能放置在其购买的虚拟机上. 值得注意的是，任务的放置不仅跨虚拟机，而且跨物理服务器. 对于网络的竞争使用不仅存在于同一虚拟机的不同任务上，更存在于同一物理计算机上的不同虚拟机中. 为简化模型描述且不失一般性，我们假设同一应用的虚拟机放置在不同的物理机，且一台虚拟机仅容纳一个任务①.

优化目标. 任务的放置决定了后续带宽分配策略可以获得的最佳带宽隔离保证. 带宽隔离保证在租户层面计算，指的是租户虚拟机间所能获得的与其需求成比例的最小带宽保证，以便其可以估计最坏情况下的网络性能. 而对网络的共享实际发生在物理机上，因此我们定义变量分别表示租户 K 在物理机 p 上聚合上行和下行带宽需求，即：

$$D_{K,p}^{\text{netOut}}=d_{i,K}^{\text{netOut}}X_{i,K}^{l,p} \tag{6}$$

$$D_{K,p}^{\text{netIn}}=d_{i,K}^{\text{netIn}}X_{i,K}^{l,p} \tag{7}$$

从而，租户 K 的进度可以定义为其在整个网络中的最小需求满足率，即

$$M_K=\min_{p\in P}\left\{\frac{R_{K,p}^{\text{netOut}}}{D_{K,p}^{\text{netOut}}},\frac{R_{K,p}^{\text{netIn}}}{D_{K,p}^{\text{netIn}}}\right\} \tag{8}$$

其中，$R_{K,p}^{\text{netOut}}$ 和 $R_{K,p}^{\text{netIn}}$ 是在物理机 p 上分配给租户 K 的聚合带宽. 隔离保证则定义为全部租户中最低的进度，即 $M^*=\min_K M_K$.

自然地，任务放置的目标是最大化隔离保证，即 $\max M^*$. 根据式(8)可知，$R_{K,p}^{\text{netOut}}$ 和 $R_{K,p}^{\text{netIn}}$ 是在任务放置确定后根据第二阶段带宽共享模型而确定的，故这里不予考虑；$D_{K,p}^{\text{netOut}}$ 和 $D_{K,p}^{\text{netIn}}$ 则依赖于任务放置的具体位置而不同. $D_{K,p}^{\text{netOut}}$ 和 $D_{K,p}^{\text{netIn}}$ 越小，则租户所获的 M_K 越大. 同时，由于我们优化的是全网络的最小带宽隔离保证，而最小带宽往往发生在网络的瓶颈链路上. 因此该优化目标即转化为，如何选择一种任务放置策略，使得网络瓶颈链路上的带宽需求最少. 形式化地，我们定义物理机的上行及下行聚合带宽需求如下：

$$D_p^{\text{netOut}}=\sum_K \omega_K D_{K,p}^{\text{netOut}} \tag{9}$$

$$D_p^{\text{netIn}}=\sum_K \omega_K D_{K,p}^{\text{netIn}} \tag{10}$$

其中，ω_K 是租户 K 的权重，云提供商可以根据租赁价格等的不同为租户分配不同的权重. 从而瓶颈链路可确定为

$$p^*=\arg\max_p \max(D_p^{\text{netOut}},D_p^{\text{netIn}}) \tag{11}$$

因此，任务放置的优化目标为

$$\min \max_p \max(D_p^{\text{netOut}},D_p^{\text{netIn}}) \tag{12}$$

约束条件. 首先，任务必须部署且唯一部署在一台虚拟机上，且租户 K 的任务只能部署在其租赁的虚拟机上：

$$\sum_l X_{i,K}^{l,p}=1,\ l\in L_K\ \forall i,K \tag{13}$$

其次，任何给定虚拟机上的累积计算及存储资源使用量都不能超过其容量：

$$\sum_i d_{i,K}^{\text{cpu}}X_{i,K}^{l,p}\le c_{K,l,p}^{\text{cpu}},\ \forall l,K,p \tag{14}$$

$$\sum_i d_{i,K}^{\text{mem}}X_{i,K}^{l,p}\le c_{K,l,p}^{\text{mem}},\ \forall l,K,p \tag{15}$$

需要注意的是，网络为弹性资源，这里我们对其不做容量限制. 同时为解决物理机和虚拟机容量异构化问题，我们采取标准化的资源需求表示. 具体而言，任务的计算及存储资源需求通过虚拟机的计算存储容量标准化，任务的上行和下行带宽需求通过物理服务器的链路容量标准化. 因此，资源需求向量中的最大分量等于 1.

最后根据 DRF，为实现优势资源公平性，任一租户在所有物理机上所获得的带宽都不能超过最佳带宽隔离保证. 因此任务放置问题可建模如下：

① 在该假设下同一租户的任务位于不同的物理机上，本文仅考虑物理机外部链路带宽分配. 但是通过修改带宽需求的表达，本文的模型和算法仍然可以支持同一租户的不同任务放置在同一物理机上.

$$\min H \tag{16}$$

$$\text{s.t.} \sum_{l} X_{i,K}^{l,p}=1,\ l\in L_K\ \forall i,K$$

$$\sum_{i} d_{i,K}^{\text{cpu}} X_{i,K}^{l,p}\le 1,\ \forall l,K,p$$

$$\sum_{i} d_{i,K}^{\text{mem}} X_{i,K}^{l,p}\le 1,\ \forall l,K,p$$

$$\sum_{K} D_{K,p}^{\text{netOut}}\le H,\ \forall p$$

$$\sum_{K} D_{K,p}^{\text{netIn}}\le H,\ \forall p$$

其中，H 表示瓶颈链路带宽需求，即

$$H=\max_{p}\max(D_{p}^{\text{netOut}},D_{p}^{\text{netIn}}) \tag{17}$$

4.3 带宽分配模型

给定特定的任务放置，我们需要相应地决定如何在不同租户的任务之间分配带宽. 理想的带宽分配策略的目标应是在当前任务放置下最大化所有租户的进度，并在优势资源公平性限制下尽可能提高网络利用率. 具体地本文采用两阶段的带宽分配模型，这种两阶段的分配机制可确保最大程度地提高利用率，同时最大化跨租户的隔离保证.

我们区分租户两种类型的带宽分配：保证带宽(Guaranteed Bandwidth)和实际带宽(Actual Bandwidth)，其中保证带宽确保实现隔离保证，实际带宽确保在隔离保证的限制下，依据最大最小公平性分配空余带宽以实现网络高利用率. 我们定义 $G_{K,p}^{\text{netOut}}$ 和 $G_{K,p}^{\text{netIn}}$ 为任务 i 所获得的上行和下行保证带宽；$R_{K,p}^{\text{netOut}}$ 和 $R_{K,p}^{\text{netIn}}$ 为任务 i 所获得的实际带宽.

首先，我们为租户分配保证带宽以确保实现隔离保证. 由前一小节可知，通过求解任务放置模型，我们得到瓶颈链路上的最小带宽需求，记为 H^*，并证明了按此放置的任务可在带宽分配中获得最优租户进度，即

$$M^*=\frac{1}{H^*} \tag{18}$$

进一步地，根据式(8)，保证带宽可计算为

$$G_{K,p}^{\text{netOut}}=M^*\times D_{K,p}^{\text{netOut}} \tag{19}$$

$$G_{K,p}^{\text{netIn}}=M^*\times D_{K,p}^{\text{netIn}} \tag{20}$$

通过预留这些保证带宽，我们可以确保隔离保证的实现. 实际上，许多目前研究证明只分配保证带宽往往会造成网络的低利用率. 因此在第二阶段我们期望能够通过分配上一阶段剩下的空闲带宽，提高网络利用率. 但是基于优势资源公平性的限制，空闲带宽分配后租户得到的总带宽(即实际带宽)不能超过租户的进度，即必须满足：

$$R_{K,p}^{\text{netOut}}\le M^* \tag{21}$$

$$R_{K,p}^{\text{netIn}}\le M^* \tag{22}$$

同时，任何给定物理机上的累积上行带宽和下行带宽分配总量不能超过其容量：

$$\sum_{K} R_{K,p}^{\text{netOut}}\le 1,\ \forall p \tag{23}$$

$$\sum_{K} R_{K,p}^{\text{netIn}}\le 1,\ \forall p \tag{24}$$

考虑到带宽分配的目标是尽可能最大化网络利用率，则该问题可建模如下：

$$\max \sum_{p}\sum_{K} R_{K,p}^{\text{netOut}}+\sum_{p}\sum_{K} R_{K,p}^{\text{netIn}} \tag{25}$$

$$\text{s.t.}\ (18)\sim(24)$$

4.4 模型分析

上述任务放置和网络分配的联合优化模型包括两个相互依赖的阶段：租户的任务放置模型决定了后续带宽分配策略所能够获得的最优进度；而带宽分配模型决定如何在不同租户的任务之间分配带宽以获得该最优进度，同时尽可能最大化网络利用率. 由于 0-1 变量的存在，式(16)中的任务放置问题是整数规划问题. 具体地该问题属于多资源瓶颈分配问题[19]且是 NP 难的. 通过松弛变量，我们可以将其转化为经典的线性规划问题，并可使用现有的线性规划求解器轻松解决，例如 Gurobi①、CVX② 等. 幸运的是任务可选的放置虚拟机集合并不大，除此之外我们可以根据剩余带宽选择最多前 l 个机器，从而进一步地降低算法执行时间. 在给定任务放置后，带宽分配模型的计算较为容易，保证带宽只需按照式(19)、(20)计算，剩余带宽则在满足优势资源公平性的前提下，按照最大最小公平性分配给租户.

5 基于应用感知的网络共享方法

在本节中，我们首先介绍基于应用感知的租户网络共享整体架构及其设计选择，然后给出任务放置和带宽分配算法的细节.

5.1 整体框架及其设计选择

本文提出的基于应用感知的网络共享方法可直接部署在现有集群管理架构内. 如图 3 所示，该机制的主要框架可分为三部分：租户通过应用程序接口定期更新其需求向量，中央控制器根据其获取到的租户资源需求、当前物理机资源状态等信息，计算任

① Gurobi Optimization. http://www.gurobi.com/
② CVX Research. http://cvxr.com/cvx/

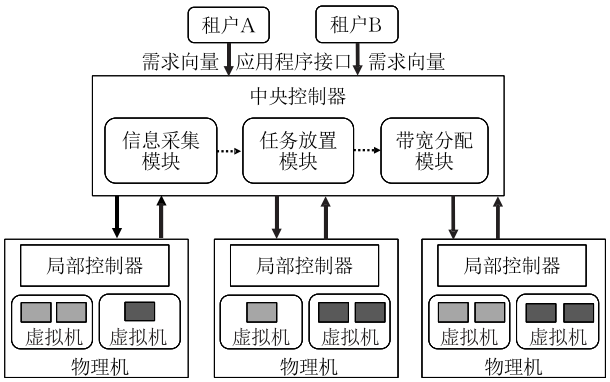


图 3　系统整体框架

务放置和网络带宽分配决策，同时负责将任务部署在相应虚拟机上，并将带宽分配决策下发给局部控制器，由局部控制器执行具体的速率控制.

应用程序接口. 租户通过应用程序接口将其资源需求提交给中央控制器，现有的集群调度器广泛地支持租户明确提交其计算和存储资源需求，这里我们主要讨论租户网络资源需求接口.许多应用程序，例如 MapReduce 类型的应用程序，其流量需求往往是高度规则且周期性的，因此这类应用程序的流量需求很好获得.除这类应用程序之外，对于产生不均匀流量的应用程序，很多现有研究方法都可以在细尺度时间上准确地估计和预测应用程序的流量矩阵[7,20-22].例如，Cicada[20]为租户提供了时间和空间上变化预测保证并着重于预测未来的流量矩阵，Proteus[21]在细尺度时间上给出了特定 MapReduce 作业的可预测的阶段性行为，并着重于对流量需求的时变性质进行建模.同时，当前的许多应用程序都具有软实时性，用户请求应在指定的等待时间目标内得到满足，从而限制应用程序的流量传输延迟.根据该数据传输延迟，流量需求矩阵可进一步转化为租户的带宽需求.由于流量工程中的现有技术在预测流量矩阵及带宽需求方面具有良好的准确性，因此本文不涉及应用程序带宽需求预测，我们假设租户正确地了解其带宽需求并提交给中央控制器.

中央控制器. 在获得租户需求向量之后，中央控制器根据其所掌握的当前所有物理机及虚拟机可用计算和存储资源，确定是否接受租户的应用程序部署请求.对允许部署的应用程序，中央控制器首先运行任务放置算法以确定每个任务应分配给哪台虚拟机，然后运行带宽分配算法以计算每个任务可以使用的带宽量.同时中央控制器负责将任务部署在相应虚拟机上，并将带宽分配决策下发给局部控制器.值得注意的是，每当有新的应用程序到来或离开，中央控制器都要更新全局资源需求向量和资源容量向量，并重新计算任务放置和带宽分配.

局部控制器. 局部控制器位于物理机上，一方面运行监控程序，监控其上所有虚拟机的运行状态和各类资源信息，并将其更新给中央控制器；另一方面，局部控制器负责该物理服务器上所有任务发送流量的速率.事实上，如果仅仅给出隔离保证的带宽量，而不在主机端进行速率限制，则通过同一链路的所有数据流仍将平均共享可用带宽.因此局部控制器的一大功能是实现主机端的速率限制，以确保所有租户实际使用的带宽不能超过 M^*.

5.2　基于应用感知的网络分享方法整体算法设计

中央控制器周期性地执行以下操作：更新全局租户资源需求向量，更新物理机及虚拟机资源容量向量，为租户的任务选择一组虚拟机，为任务分配网络带宽.算法 I 给出了基于应用感知的网络分享的整体操作过程.一系列事件都会触发这些操作，例如新租户的加入、租户所有任务执行完毕并退出系统、租户改变其需求向量等.

若新租户加入，当前系统中的资源需求向量改变，如果中央控制器重新执行过程 I，则可能会找到新的瓶颈链路及其上的带宽需求，从而造成当前任务部署虚拟机发生变化.然而，任务迁移会造成极大的开销，我们倾向于不改变正在执行的任务位置.因此，在不改变当前的瓶颈链路及其上的带宽需求的前提下，将新租户的任务部署在其他空闲虚拟机上，并根据原来的最佳进度限制其带宽分配量，通过这样的操作既不改变打断当前应用程序的执行，同时也为新的应用程序分配足够的带宽.但是若当前网络处于过载情况下，则可能无法找到合适的放置而不改变瓶颈链路带宽需求，在这种情况下我们重新执行过程 I；若租户应用程序完成，这部分任务释放其占用的所有计算、存储和网络资源，从而改变当前系统中的资源需求向量，虚拟机可用资源容量以及物理机可用网络资源容量，这种情况下中央控制器重新执行过程 I；若租户的资源需求发生改变，从而当前系统中的资源需求向量改变，这种情况下我们单纯地采取重新执行过程 I.通过上述的操作过程，我们可以轻松地处理系统动态性并获得良好的性能.

过程 I.　基于应用感知的网络共享方法操作过程.

1. 初始化当前系统中租户的资源需求向量
2. 监控并获取当前系统中虚拟机可用计算和存储资源容量 $\boldsymbol{c}_{K,p,l}$

3. 监控并获取当前系统中物理机可用网络资源容量 $\boldsymbol{c}_p$
4. 将物理机按其可用资源容量倒序排列
5. 选取排列前 t 的物理机为候选物理机集
6. 选取候选物理机上所有属于租户 K 的虚拟机作为该租户的候选虚拟机集
7. 计算最佳瓶颈带宽需求并执行任务布置
   //调用算法 I
8. 计算租户的保证带宽
   //调用算法 II 中的计算保证带宽过程
9. 计算租户的实际带宽
   //调用算法 II 中的计算实际带宽过程

5.3 任务放置和带宽分享

在本节中,我们将详细介绍任务放置和带宽分配的算法设计.给定不同租户的应用程序集合以及有关任务的预期带宽需求和其他约束(如 CPU)信息,我们首先将这些任务分配给一组虚拟机,然后为它们分配带宽以在满足最优进度限制的前提下实现较高的利用率.

在第一阶段,即任务放置阶段,给定租户的需求向量以及当前系统中的可用资源容量,通过问题求解我们希望获得最低的瓶颈链路带宽需求,从而确保得到所有租户的最优进度.若我们将不同物理机的上行带宽和下行带宽看作不同的资源,根据 DRF[1] 中的定义,为满足优势资源公平性所有租户的进度都必须相等,也就是说该最优进度即为隔离保证,也就是,

$$M^*=\frac{1}{H^*}.$$

通过之前的证明,我们发现能够获得这样的瓶颈链路带宽需求的任务放置是最佳的.通过松弛变量法,我们将该 NP 难的任务放置问题转化为经典的线性规划问题,从而可以使用现有的线性规划求解器解决.具体地,我们采用贪婪算法求得该问题的近似解,如算法 I 所示.

算法 I. 第一阶段任务放置算法.

输入:租户 K 的计算和存储需求 $d_{i,K}^{\mathrm{cpu}}$ 和 $d_{i,K}^{\mathrm{mem}}$ $\forall i,K$,以及其带宽需求向量($\boldsymbol{D}_{K,p}^{\mathrm{netOut}}$,$\boldsymbol{D}_{K,p}^{\mathrm{netIn}}$),$\forall K$

输出:瓶颈链路带宽需求 H^*,租户 K 的任务放置向量 $\boldsymbol{X}_{i,K}^{l,p}$

1. 通过松弛变量将问题(16)转化为线性规划问题
2. 采用线性规划求解器求解转换后的问题
3. 计算得出瓶颈链路的最小带宽需求 H^*
4. FOR 租户 K 的所有任务 DO
5. 初始化放置向量 $\boldsymbol{X}_{i,K}^{l,p}=0,\forall l$
6. 选择拥有最大 $\boldsymbol{X}_{i,K}^{l,p}$ 值的虚拟机
7. 记该选择的虚拟机为 l^*,并设 $X_{i,K}^{l^*,p}=1$

   END FOR
8. 更新 $\{\boldsymbol{X}_{i,K}^{l,p}\}\cup X_{i,K}^{l^*,p}$
9. 根据 $\{\boldsymbol{X}_{i,K}^{l,p}\}$ 部署任务

算法 II. 第二阶段带宽分配算法.

输入:瓶颈链路带宽需求 H^*,租户 K 的带宽需求($\boldsymbol{D}_{K,p}^{\mathrm{netOut}}$,$\boldsymbol{D}_{K,p}^{\mathrm{netIn}}$),$\forall K$,租户 K 的任务放置向量 $\boldsymbol{X}_{i,K}^{l,p}$

输出:租户 K 的保证带宽分配($\boldsymbol{G}_{K,p}^{\mathrm{netOut}}$,$\boldsymbol{G}_{K,p}^{\mathrm{netIn}}$),$\forall K$ 和实际带宽分配($\boldsymbol{R}_{K,p}^{\mathrm{netOut}}$,$\boldsymbol{R}_{K,p}^{\mathrm{netIn}}$),$\forall K$

1. 计算最佳进度 $M^*=1/H^*$
2. FOR 全部 $\boldsymbol{X}_{i,K}^{l,p}=1$ DO
3. $G_{K,p}^{\mathrm{netOut}}=M^*\times D_{K,p}^{\mathrm{netOut}}$
4. $G_{K,p}^{\mathrm{netIn}}=M^*\times D_{K,p}^{\mathrm{netIn}}$

   END FOR
5. FOR 全部物理机 DO
6. 将物理机的剩余带宽按最大最小公平性分配给租户
7. 确保实际带宽不超过最优进度,即满足 $(R_{K,p}^{\mathrm{netOut}},R_{K,p}^{\mathrm{netIn}})\leq M^*,\forall K,p$

   END FOR

在第二阶段,给定任务的放置以及租户理论上可以获得的最优进度,本文提出的带宽分配算法在确保优势资源公平性限制下,试图提高网络整体的利用率.首先,我们根据式(19)和(20)获得租户的保证带宽.保证带宽确保租户获得其最佳隔离保证,并实现租户间的优势资源公平性.然而若仅按照该保证带宽分配,系统的整体网络利用率会较低.同时,与计算和存储资源不同的是,网络资源是弹性的,这意味着若分配的带宽多于任务需求,会提高其数据传输速度,从而降低任务执行时间.基于此,为提高网络利用率和任务传输速率,我们按照最大最小公平性将空余的带宽分配给租户,并确保租户实际获得的带宽不超过 M^*,即

$$(G_{K,p}^{\mathrm{netOut}},G_{K,p}^{\mathrm{netIn}})\leq(R_{K,p}^{\mathrm{netOut}},R_{K,p}^{\mathrm{netIn}})\leq M^*.$$

这意味着空余资源的分配采用最大最小公平性在当前物理机的租户之间分配,但受限于 M^*.值得注意的是,我们只考虑网络带宽在租户之间的分配,租户内部的带宽如何分配由租户自行决定,例如采用公平分配、最短优先流调度等.

6 实验结果分析

在本节中,我们将通过实验评估本文提出的基于应用感知的网络多租户共享方法,该实验涵盖了广泛的网络条件和实际负载.

6.1 实验设置

为了在更大范围内和更多参数选择下评估本文

的基于应用感知的网络分享方法，我们构建了事件驱动的集群模拟器.类似真实的集群环境，该模拟器能够模拟以下方面：租户应用程序到达或离开，任务计算和物理资源需求，租户应用程序的带宽需求矩阵等.此外，我们采用与真实集群相同的物理机配置文件，每台物理机具有 12 CPU，32 GB 内存，1 Gbps 上行及下行带宽，并假设每台物理机托管 4 台虚拟机，每台虚拟机具有相同的计算和存储资源容量，也就是(3 CPU，8 GB 内存).

我们假设租户应用程序的到达符合泊松分布，应用程序内具有多对多的通信模式.为使实验评估尽可能模拟真实的集群环境，我们构建多种不同类型的应用程序.其中应用程序的大小，即其所包含的任务数量有 25、50、75、100 四种，任务具有高内存(8 GB)需求和低内存(2 GB)需求两种，以及 1 CPU、2 CPU 和 3 CPU 三种计算需求.我们将每个流的大小随机设置为 200 MB 至 400 MB.任务拥有的虚拟机数量与其应用程序的任务数量成正比.基于系统状态的全局视图，包括全局资源需求向量和可用资源容量，中央控制器计算任务的最佳放置并为所有应用程序的流分配带宽.每当新的应用程序到达或离开，模拟器都将更新全部应用程序的任务之间剩余传输数据量和带宽需求，并触发中心控制器计算最佳任务布置和带宽分配.任务在此带宽分配量下传输其数据，直至该任务剩余传输数据量为 0(即该任务数据传输完成)，或新的应用程序到达或离开等事件触发中心控制器重新计算.

比较方案：我们将本文提出的网络共享方法(图中用 ANS 表示)与当前最新带宽分配算法进行比较，分别是每流公平共享算法(Per-flow Fairness)、优势资源公平共享算法(Dominant Resource Fairness，DRF)和高利用率保证算法(High Utilization with Guarantees，HUG).其中 Per-flow Fairness 将带宽平均分配给所有流，没有任何速率限制，DRF 和 HUG 根据不同的规则限制每个流所能使用的带宽.需要注意的是，这些比较算法不涉及虚拟机或任务的放置，只专注于网络带宽的分享.为能够更好地进行比较，在这些比较方案中我们采用轮询的任务放置.

指标.本文提出的网络分享方法旨在提高带宽隔离保证，网络利用率和租户应用程序性能.对于前两个指标，我们计算提高(或减少)的百分比为

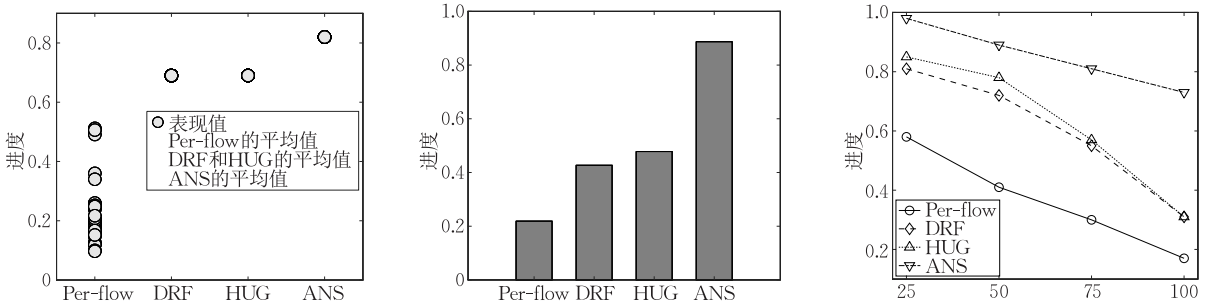
$$\frac{\text{本文方法的指标值}-\text{比较方案的指标值}}{\text{比较方案的指标值}}\times 100\%.$$

例如 25%的提高意味着本文的方法 1.33 倍好于比较方案.为量化租户的应用程序性能，我们定义与最小完成时间相比，本文方法和比较方案的完成时间减慢比率，其计算方式将在 6.4 节介绍.

6.2　租户进度的实验结果

根据第 4 节的分析可知，租户所获得的进度是其在上行或下行瓶颈链路上分配到的带宽量(即进度最多为 1 Gbps).图 4(a)给出了一次实验中租户的应用程序进度分布，这里我们不考虑租户应用程序到达或离开，即离线状态下根据不同算法获得的租户进度分布.我们发现 Per-flow Fairness 呈现出广泛的进度变化，该方法将带宽在所有流之间平均分配，因此应用程序内拥有较多数据流的租户会分配得到更多的带宽.其次，无论是 DRF、HUG 还是本文提出的方法，所有租户都将获得一样的进度，这是由于优势资源公平性的限制，确保所有租户都能获得相同的进度以实现隔离保证.除此之外，我们还发现 DRF 和 HUG 获得近乎相同的进度值，这是因为 HUG 直接采用了优势资源公平性来最大程度地提高隔离保证.同时本文的方法明显地获得了比 HUG 更高的进度，具体地在本次实验中我们的方法相比 HUG 的进度提高了 1.19 倍.这种差距的原因在于，DRF 和 HUG 是在给定任务放置下获得最佳进度，其算法本身不涉及任务的放置.但本文所提出的基于任务放置的网络共享方法通过巧妙地部署任务，使得带宽需求在全网络中尽可能平均，因此进度的提高是显而易见的.

进一步地，图 4(b)给出了 Per-flow fairness、DRF、HUG 和我们的方法在多次实验中获得的平均进度值.与图 4(a)所展示的瞬时进度不同，图 4(b)给出了长期进度的平均值.与瞬时进度不同，DRF 和 HUG 在长期运行下所获得的进度效果不佳，DRF 的平均进度值为 0.427，HUG 的平均进度值为 0.478.相反，我们的方法可以获得的平均进度值约为 0.887，具体地，与 DRF 相比本文方法的进度提高百分比约为 $\frac{0.887-0.427}{0.427}\times 100\%=107.7\%$，与 HUG 相比进度提高百分比约为 85.6%.图 4(c)给出了随着网络需求总量的变化，不同方法获得的进度值的变化.我们发现当应用程序越大，即拥有的任务数量越多，网络需求量越大，所有方法的进度值随之降低，但我们的方法始终能够获得更高的进度.

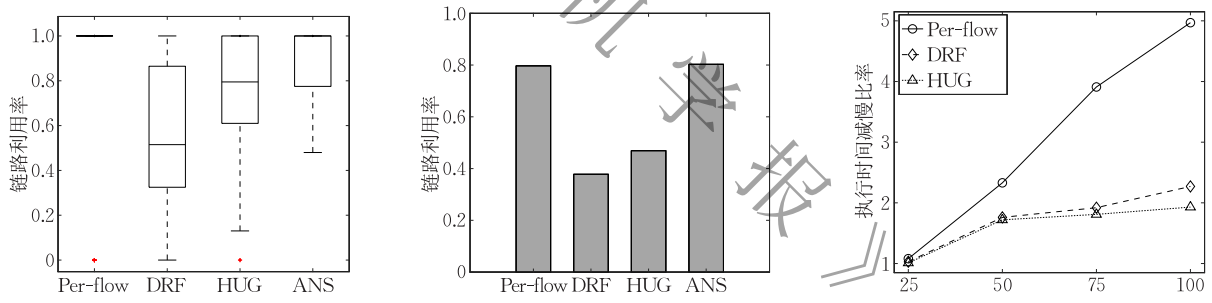


(a) 一次实验中租户进度的分布 (b) 长期实验中租户进度的平均值 (c) 租户进度随应用程序总需求变化的情况

图 4 租户进度的实验评估结果

6.3 网络利用率的实验结果

在带宽分配阶段我们期望最大化网络利用率，同时确保租户的隔离保证. 高的网络利用率意味着租户的应用程序可以更快地完成，从而有更多的空余带宽供其他应用程序共享. 图 5(a)表示的是链路利用率的分布. 图 5(b)表示的是 Per-flow Fairness、DRF、HUG 和我们的方法在多次实验中获得的平均网络利用率. 我们发现 DRF 具有最差的网络利用率，这是由于优势资源公平性的限制，在一些极端情况下其资源利用率甚至可能降为 0. HUG 的资源利用率较 DRF 好，这是因为 HUG 在 DRF 的优势资源公平性限制下，考虑弹性资源从而最大程度地提高资源利用率. 相反地，Per-flow Fairness 的网络利用率却是很高的，这是因为链路带宽在所有流之间平均分配，并会用尽所有带宽，因此具有较大的网络利用率. 同时我们的方法能够获得与 Per-flow Fairness 类似的利用率，远高于 HUG 和 DRF 的利用率，这得益于我们在第一阶段优化了任务布置使得全网络上的带宽需求尽可能地平均，这在提高进度的同时也相应提高了网络利用率，进一步地我们同时也采用最大最小公平性分配剩余带宽.



(a) 链路利用率的分布 (b) 链路利用率的平均值 (c) 链路利用率随应用程序总需求变化的情况

图 5 网络利用率的实验评估结果

同时，实验中我们发现网络利用率与应用程序特征具有较强的相关性. 图 5(c)给出了网络利用率随着应用程序大小的变化. 随着应用程序内任务数量的增加，网络利用率随之增大，但四种方案的增大比率不一样. 其中增幅最明显的是 Per-flow Fairness，这是因为链路上只要有需求，该方案都会将所有带宽分配，而任务数量的增加会提高带宽需求，从而带来较大的利用率提高. 与此同时，DRF、HUG 和本文方法在任务数量增加的同时，利用率的增加较平缓. 这是因为这三种方案在分配带宽时都遵循优势资源公平性的限制，但 DRF、HUG 和本文方法依然有差别. DRF 和 HUG 中任务的放置是以轮询的方式进行，因此瓶颈链路以及其上的带宽需求会随之变化，随着整体的租户网络带宽需求的增加，瓶颈链路上的需求也会相应增加，但是出于优势资源公平性的限制，空余资源的使用也会受到限制. 相反地，本文提出的方法随着任务数量的增加，其利用率一直维持较好的水平，即使在低租户需求的网络中. 这种性能的获得依赖于我们方法中第一阶段的任务放置优化.

6.4 应用程序性能的实验结果

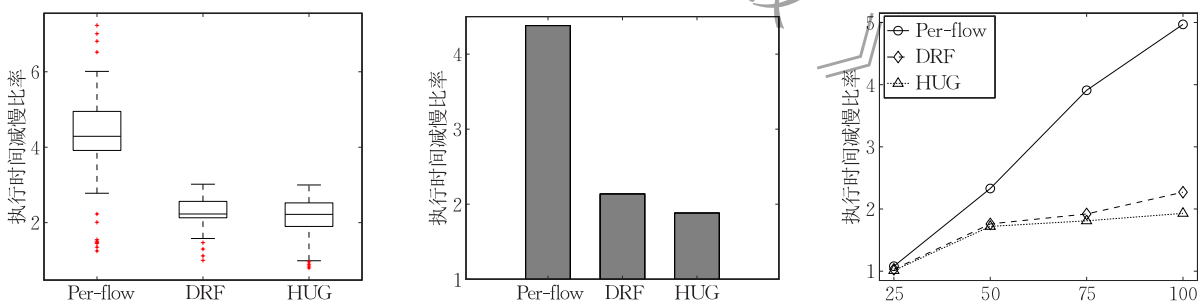
应用程序性能是衡量网络分配方案的长期指标. 瞬时的高利用率或高进度并不能保证应用程序获得良好的性能，只有在长期的过程中保证高利用率才有利于应用程序的执行. 这里我们以应用程序执行时间作为其性能指标. 应用程序执行时间为应

用程序从到达直至离开所经历的完整时间，由计算时间和数据传输时间两部分组成. 其中，计算时间与一系列复杂因素相关，如数据局部性等，超出本文研究范畴. 不失一般性地，这里模拟器简单预设计算时间与其计算存储资源需求成正比；传输时间是所有任务完成数据传输所需的总时间，由任务的数据传输量和所分配到的带宽所决定. 由于每次应用程序到达或离开等事件都会触发模拟器重新进行任务布置和带宽分配. 因此随着应用程序动态地到达或离开，任务所需传输的数据量和其获得的带宽分配也会不断地发生变化. 因此，数据传输是一个长期的过程，应用程序的传输时间由该应用程序内所有数据流中最慢的传输决定.

具体地，我们采用执行时间减慢比率作为衡量指标，并将其定义为通过本文方法的执行时间标准化的比较方案的执行时间，即若执行时间减慢比率大于 1，则说明我们的方法执行时间更短，应用程序性能更好；若执行时间减慢比率小于 1，则说明比较方案的应用性能更好. 该定义公式为

$$\text{执行时间减慢比率}=\frac{\text{比较方案的执行时间}}{\text{本文方法的执行时间}}.$$

图 6(a)表示的是应用程序执行时间的分布. 图 6(b)给出了 Per-flow Fairness、DRF 和 HUG 的平均执行时间减慢比率. 我们发现所有比较方案的该比率都大于 1，证明本文方法获得最短的应用程序执行时间. 具体地，Per-flow Fairness 的执行时间最长，这也证明了流层面的高网络利用率并不能确保应用程序层面的高性能，事实上只有租户应用程序内所有的流都尽快完成传输，才会提高应用程序的性能. 我们进一步发现 Per-flow Fairness 中应用程序执行时间有较大的波动，这是因为该方案不对分享公平性进行限制. 而 HUG 比 DRF 获得更好的应用程序性能，两者都遵循优势资源公平性的限制，但是 HUG 通过考虑弹性资源将空余的带宽分配尽可能多地分配给应用程序，从而降低应用程序执行时间. 同样地，全网络的需求会对所有方案的应用程序执行时间造成影响. 一般地，网络负载越高，应用程序需求越大，单个应用程序获得的带宽越少，造成执行时间越长. 图 6(c)给出了应用程序执行时间减慢比率随着应用程序大小的变化而变化的情况. 具体地，我们发现 Per-flow Fairness 的执行时间对任务数量的增加最为敏感，在任务数量为 25 时，网络负载最低，四种方案的应用程序执行时间相差最小，但当任务数量增长到 75 时，网络出现轻微过载，Per-flow Fairness、DRF 和 HUG 的执行时间减慢比率显著增加，但是此时我们的方法依然能够通过平衡网络各链路的负载，获得较好的应用程序性能. 当任务数量增长到 100 时，这种差距继续增加，证明我们的方法可以很好的平衡网络需求，提高应用程序性能. 通过上述实验，我们可以得出基于应用感知的网络共享策略在进度、网络利用率和应用程序性能各方面都能获得优秀的性能.



(a) 应用程序执行时间的分布　(b) 平均应用程序执行时间比较　(c) 应用程序执行时间随其需求变化的情况

图 6　应用程序执行时间的实验评估结果

7　总　结

近年来，针对多租户的数据中心网络共享研究已成为数据中心资源管理和分配的重要领域，网络分配决策会对应用程序端到端的性能产生直接且重要的影响. 为了衡量租户的整体数据传输率，本文引入进度(Progress)的指标. 该指标被定义为租户在所有链路上经需求标准化后的最小带宽分配量，反映的是租户能够完成其数据传输的最慢速率. 通过最大程度地提高租户进度，能够大幅提高上层应用程序的性能. 进一步地，本文观察发现任务放置位置的不同会严重影响租户所获得的进度，通过理论分析我们证明获得所有租户最优进度的关键在于最小化网络瓶颈链路上的带宽需求. 本文提出的基于应用感知的网络共享方法包括两个相互依赖的阶段：

任务放置优化以决定任务在不同虚拟机上的放置，进而决定后续带宽分配所能够获得的最优进度；带宽分配策略决定如何在不同租户的任务之间分配带宽以获得该最优进度，同时最大化网络利用率. 实验证明我们提出的方法可以确保租户获得长期的高进度，高应用程序性能以及网络利用率.

参 考 文 献

[1] Li Dan, Chen Gui-Hai, Ren Feng-Yuan, et al. Data center network research progress and trends. Chinese Journal of Computers, 2014, 37(2): 259-274(in Chinese)
(李丹，陈贵海，任丰原等. 数据中心网络的研究进展与趋势. 计算机学报，2014，37(2)：259-274)

[2] Liu Shi-Hai, Sun Yu-Qing, Liu Gu-Yue. An adaptive bandwidth allocation algorithm for virtual machine migration based on service features. Chinese Journal of Computers, 2013, 36(3): 1816-1825(in Chinese)
(刘诗海，孙宇清，刘古月. 面向业务特征的自适应虚拟机迁移带宽分配算法. 计算机学报，2013，36(8)：1816-1825)

[3] Ghodsi A, Zaharia M, Hindman B, et al. Dominant resource fairness: Fair allocation of multiple resource types//Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI). Boston, USA, 2011: 323-336

[4] Chowdhury M, Liu Z, Stoica I. HUG: Multi-resource fairness for correlated and elastic demands//Proceedings of the 13th USENIX Conference on Networked Systems Design and Implementation (NSDI). Santa Clara, USA, 2016: 407-424

[5] Shieh A, Kandula S, Greenberg A, et al. Seawall: Performance isolation for cloud datacenter networks//Proceedings of the 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud). Boston, USA, 2010: 1

[6] 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 (CoNEXT). Philadelphia, USA, 2010: 1-12

[7] Ballani H, Costa P, Karagiannis T, et al. Towards predictable datacenter networks//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). Toronto, Canada, 2011: 242-253

[8] Rodrigues H, Santos J R, Turner Y, et al. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks//Proceedings of the 3rd Conference on I/O Virtualization (WIOV). Portland, USA, 2011: 6

[9] Lam V T, Radhakrishnan S, Vahdat A, et al. NetShare and stochastic: Predictable bandwidth allocation for data centers. Computer Communication Review, 2012, 42(3): 6-11

[10] Popa L, Kumar G, Chowdhury M, et al. FairCloud: Sharing the network in cloud computing//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). Helsinki, Finland, 2012: 187-198

[11] Shieh A, Kandula S, Greenberg A, et al. Sharing the data center network//Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI). Boston, USA, 2011: 309-322

[12] Mogul J, Popa L. What we talk about when we talk about cloud network performance. ACM SIGCOMM Computer Communication Review, 2012, 42(5): 44-48

[13] Xie J, Guo D, Zhu X, et al. Minimal fault-tolerant coverage of controllers in IaaS datacenters. IEEE Transactions on Service Computing, 2017: 1-1, DOI: 10.1109/TSC.2017.2753260

[14] Xie J, Guo D, Wu J, et al. Exploiting reliable and scalable multicast services in IaaS datacenters. IEEE Transactions on Service Computing, 2018: 1-1, DOI: 10.1109/TSC.2018.2877733

[15] Jeyakumar V, Alizadeh M, Mazieres D, et al. EyeQ: Practical network performance isolation at the edge//Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI). Lombard, USA, 2013: 297-312

[16] Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). Seattle, USA, 2008: 63-74

[17] Guo C, Lu G, Li D, et al. BCube: A high performance, server-centric network architecture for modular data centers //Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). Barcelona, Spain, 2009: 63-74

[18] Al-Fares M, Radhakrishnan S, Raghavan B, et al. Hedera: Dynamic flow scheduling for data center networks//Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI). San Jose, USA, 2009: 19

[19] Pentico D W. Assignment problems: A golden anniversary survey. European Journal of Operational Research, 2007, 176(2): 774-793

[20] LaCurts K, Mogul J C, Balakrishnan H, et al. Cicada: Introducing predictive guarantees for cloud networks// Proceedings of the 6th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud). Philadelphia, USA, 2014: 14

[21] Xie D, Ding N, Hu Y C, et al. The only constant is change: Incorporating time-varying network reservations in data centers//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). Helsinki, Finland, 2012: 199-210

[22] 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 (IMC). Melbourne, Australia, 2010: 267-280

LI Yan, Ph. D. candidate. Her current research interests include data center network and cloud computing.

GUO De-Ke, Ph. D., professor, Ph. D. supervisor. His current research interests include network computing and systems, distributed computing and systems, and mobile computing.

CAO Xiao-Feng, Ph. D. candidate. His current research interests include edge computing and big data analysis.

CHEN Hong-Hui, Ph. D., professor, Ph. D. supervisor. His current research interests include information system, cloud computing and information retrieval.

Background

In the infrastructure-as-a-service (IaaS, i. e., cloud) data center, tenants deploy applications on their virtual machines (VMs) rent from the cloud. This strategy allows tenants multiplex compute and storage resources with isolation guarantee; on the contrary, network resources are shared among tenants in a "best-effort" manner. Therefore, applications of different tenants compete for the shared network, and their allocated bandwidth is limited by a series of complex factors, such as system load, VM placement, and oversubscription ratio. This caused significant differences in network performance among tenants, which severely impaired application performance, not only made tenant costs unpredictable but also caused revenue loss for cloud service providers. There is a general agreement in broad terms that a network sharing solution among tenants should be fair, predictable, and efficient. For tenants, they expect the guaranteed minimum bandwidth to enable the performance predictability; meanwhile, for cloud providers, they strive for the work conservation to achieve the high network utilization and the strategy-proofness to ensure the performance isolation. This paper tries to focus on a novel metric, progress, which is used to measure the overall data transfer rate of a tenant and indicate how fast a tenant can complete its data transfer. By maximizing the tenant's progress, application performance such as execution time can be optimized. Currently, much work has proved that given the bandwidth demand vectors of all tenants, the state-of-the-art bandwidth allocation schemes i. e., DRF and HUG, can obtain the optimal progress. However, we found that by changing the bandwidth demand distribution, the tenant's progress and isolation guarantee can be further improved. Thus we systemically propose a network sharing mechanism by exploiting the benefits of careful placement of tenants' applications. The insight is to tune and optimize the placement of applications to cause a desired traffic pattern for achieving high progress. Consequently, the tenant's progress and network utilization would be further improved. This work is supported by the National Key R&D Program of China (Grant No. 2018YFE0207600) and the Key Program of Joint Funds of the National Natural Science Foundation of China (Grant No. U19B2024).