Towards Bandwidth Guarantee for Virtual Clusters under Demand Uncertainty in Multi-tenant Clouds

Lei Yu, Haiying Shen, Zhipeng Cai, Ling Liu and Calton Pu

Abstract—In the cloud, multiple tenants share the resource of datacenters and their applications compete with each other for scarce network bandwidth. Current studies have shown that the lack of bandwidth guarantee causes unpredictable network performance, leading to poor application performance. To address this issue, several virtual network abstractions have been proposed which allow the tenants to reserve virtual clusters with specified bandwidth between the Virtual Machines (VMs) in the datacenters. However, all these existing proposals require the tenants to deterministically characterize the bandwidth demands in the abstractions, which can be difficult and result in inefficient bandwidth reservation due to the demand uncertainty. In this paper, we explore a virtual cluster abstraction with stochastic bandwidth characterization to address the bandwidth demand uncertainty. We propose Stochastic Virtual Cluster (SVC), which models the bandwidth demand between VMs in a probabilistic way. Based on SVC, we develop a stochastic framework for virtual cluster allocation, in which the admitted virtual cluster's bandwidth demands are satisfied with a high probability. Efficient VM allocation algorithms are proposed to implement the framework while reducing the possibility of link congestion through minimizing the maximum bandwidth occupancy of a virtual cluster on physical links. Using simulations, we show that SVC achieves the trade-off between the job concurrency and the average job running time, and demonstrate its effectiveness for accommodating cloud application workloads with highly volatile bandwidth demands and its improvement to work-conserving bandwidth enforcement.

Index Terms—cloud computing, bandwidth guarantee, network abstraction, virtual network, virtual machine placement.

1 Introduction

Based on modern virtualization of datacenters, cloud computing delivers cost-effective and powerful Infrastructure as a Service (IaaS) for a wide spectrum of applications like Web services and Hadoop [1]. However, in the shared and multi-tenant cloud network infrastructures, the competition of applications for the scarce network resources have caused unpredictable application performance in the cloud [2]. The lack of guaranteed network bandwidth causes variable data transmission latency and job completion time, leading to poor job scheduling and datacenter throughput [3], [4].

Recently, several virtual network abstractions [5], [6], [7], [8] have been proposed, which specify a virtual cluster (alternatively, a virtual datacenter) required by the tenants and the bandwidth requirements between VMs. They explicitly enable the tenants to accurately specify their bandwidth demands for their virtual clusters while enabling efficient bandwidth provision in datacenter networks. SecondNet [5] aims to guarantee end-to-end bandwidth for each pair of virtual machines (in short, VMs). Oktopus [6] introduces a virtual cluster model that requires a virtual topology comprising N VMs connected to a virtual switch with virtual links having bandwidth capacity specified by the tenants. Considering that many cloud applications have time-varying bandwidth requirements, a temporally-interleaved virtual cluster model TIVC [7] is proposed. It allows the tenants to specify different bandwidth demands at different time intervals. CloudMirror [8] drives the network abstraction based on the application communication structure instead of the horse model. While these methods provides efficient bandwidth reservation to the tenants, they require reliable and deterministic estimate of bandwidth demand among VMs; the bandwidth requirement is estimated based on the profiling runs and the analysis of the cloud applications.

Unfortunately, recent measurement studies [9], [10] show that the network traffic is highly volatile in production datacenters. Since the above bandwidth guarantee approaches [5], [6], [7], [8] require deterministic bandwidth specification in virtual network abstractions, they have to specify over-provisioned bandwidth to deal with the stochastic traffic. This can lead to insufficient bandwidth provision for guaranteeing application predictability, inefficient network utilization, significant resource waste to cloud providers and high costs to tenants. It has been shown that the stochastic traffic has impact on various aspects of traffic engineering, such as link utilization and bandwidth provisioning [11]. It is also a difficult task for a tenant to determine the accurate amount of bandwidth that the virtual cluster needs at the running time due to demand uncertainty. To remedy these shortfalls, this paper develops a stochastic bandwidth allocation approach for virtual clusters, which takes into account the demand uncertainty based on the stochastic characterization of the bandwidth demand.

The primary contribution of this paper is to explore a new network allocation abstraction called *Stochastic Virtual Cluster* (SVC). It extends the virtual cluster abstraction with probabilistic distributions of bandwidth demand on virtual links to capture the traffic uncertainty and variance of cloud applications. The distributional information typically can be derived by statistical procedures for forecasting network traffic from measurements [12], [13]. Such information has not been exploited in existing solutions for bandwidth guarantee.

With using the demand distribution information, SVC aims to address the insufficiency of existing virtual network abstractions for handling traffic burstiness and demand uncertainty, and is expected to improve the network utilization. Several works like Gatekeeper [14], EyeQ [15] and ElasticSwitch [16] are proposed to ad-

- *Lei Yu, Ling Liu and Calton Pu are with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA, 30303. E-mail: lyu79@gatech.edu, {ling.liu, calton.pu}@cc.gatech.edu*
- *Haiying Shen is with the Department of Computer Science, University of Virginia, Charlottesville, VA, 22904. E-mail: hs6ms@virginia.edu*
- *Zhipeng Cai is with the Department of Computer Science, Georgia State University, Atlanta, GA, 30303. E-mail: zcai@gsu.edu*

dress these issues through work-conserving rate control at runtime. They enforce the minimum bandwidth guarantee specified by the horse model through the admission control of virtual clusters and proper VM placement. The traffic burstiness is handled by dynamically increasing the rate beyond the deterministic bandwidth specification in the abstraction to achieve work-conservation. However, because these approaches use the hose model with deterministic bandwidth, they place VMs without respect to traffic burstiness and the resulting virtual cluster allocations are not optimal for stochastic bandwidth demands. The VM placement oblivious to traffic bustiness may let the bursty flows from different VM pairs go across the same link. The dynamical rate adjustment would further cause bandwidth competition among these virtual clusters and link congestion. If the VMs of virtual clusters are admitted and placed with the consideration of traffic dynamics, that is, the VMs are placed to prevent bursty traffic from different VM pairs sharing the same links, the performance gain can be increased for dynamic rate adjustment. With specifying stochastic bandwidth demand, the proposed SVC abstraction provides opportunity for us to address the traffic burstiness and improve network utilization at an earlier time, i.e., during the network planning phase conducting admission control and VM placement. Thus, while our work is orthogonal to these existing works [14], [15], [16], they can be integrated together to improve the network utilization and performance for virtual cluster allocation.

Based on SVC, we propose a stochastic framework for virtual cluster allocation, where the bandwidth demands of virtual clusters in the datacenter are satisfied with a high probability. In this framework, virtual cluster admission and VM placement take into account the effect of statistical bandwidth multiplexing with ensuring probabilistic bandwidth guarantee for all the admitted virtual clusters. Being aware of bandwidth demand uncertainty for virtual cluster allocation, this framework improves the network utilization and job concurrency in the clouds, and thus provides better performance guarantee to cloud applications with highly volatile bandwidth demands.

To implement such a framework, a fundamental problem is how to allocate VMs in a physical datacenter for an SVC abstraction while ensuring the probabilistic bandwidth guarantee for all tenants. To solve the problem, as opposed to previous approaches proposed for allocating deterministic virtual clusters [6], [7], [8], we need to characterize the bandwidth occupancy on any links for a virtual cluster and establish the condition for a valid VM allocation in a probabilistic manner. Also, we point out that previous VM allocation algorithms, although providing good locality of VMs in a virtual cluster, is not optimal in terms of the bandwidth occupancy of physical links. In this paper, we derive the condition for a valid VM allocation that provides probabilistic bandwidth guarantee. Based on that, we then propose a dynamic programming based VM allocation algorithm that finds the optimal allocation for a virtual cluster within the lowest-level subtree with the goal to minimize its bandwidth occupancy on physical links in order to reduce the possibilities of link congestion. Furthermore, we consider the SVC model with heterogeneous bandwidth demands that may follow different probability distributions, and develop a heuristic algorithm towards finding the optimal VM allocation.

In summary, the contributions of this paper are as follows.

- We introduce a new virtual cluster abstraction SVC with stochastic bandwidth demands to account for bandwidth demand uncertainty. Based on that, we propose a stochastic framework for virtual cluster allocation that aims to provide probabilistic bandwidth guarantee to tenants.
- We devise efficient VM allocation algorithms for SVC with homogeneous and heterogeneous bandwidth demands, which not only achieve good locality but also minimize the bandwidth occupancy of the virtual cluster on physical links.
- We extend SVC with minimum bandwidth guarantee such that it can be incorporated with the bandwidth enforcement approaches for work-conserving bandwidth guarantee.
- We demonstrate the effectiveness of SVC and our network sharing solution with SVC by extensive simulations. The results show SVC yields better performance to cloud applications with highly volatile bandwidth demands and achieves the trade-off between the job concurrency and average job running time, compared with previous deterministic bandwidth abstractions.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 describes our SVC model and the stochastic framework for virtual cluster allocation with probabilistic bandwidth guarantee. Section 4 and Section 5 present the VM allocation algorithms for SVC with homogeneous and heterogeneous bandwidth demand, respectively. Section 7 shows our simulation results.

2 Related Work

2.1 Network Sharing in Clouds

Given that the lack of bandwidth guarantee can significantly degrade the performance of cloud applications, several network sharing solutions for multi-tenant datacenters have been proposed [4], [5], [6], [7], [8], [17], [18]. These solutions have different policies for sharing the bandwidth, including bandwidth reservation [5], [6], [7], [8] and weight proportional bandwidth sharing [4], [17], [18].

Secondnet [5] and Oktopus [6] provide bandwidth guarantee through deterministic bandwidth reservations in the network. Secondnet [5] proposes a virtual datacenter abstraction which specifies the bandwidth requirements among each VM pair. A central controller determines the flow rate and the path for each VM-to-VM pair and inform the end hosts. Based on the hose model, Oktopus [6] proposes a virtual cluster abstraction for network reservation, in which a virtual cluster request $\langle N, B\rangle$ requires a virtual topology comprising N machines connected by links of bandwidth B to a virtual switch. TIVC [7] extends the virtual cluster model of Oktopus [6] with time-varying bandwidth reservation in order to capture the time-varying networking requirement of cloud applications. In [19], the virtual cluster model is further extended for allowing heterogeneous bandwidth requirements for different VMs. The algorithm proposed in [19] addresses the allocation of virtual clusters with heterogeneous bandwidth demand. It improves the search efficiency by dividing 2^N possible subsets of N VMs into N^2 groups and examining the allocatability of VM subsets by group. The hose model based abstractions like Oktopus and TIVC well address batch processing applications like MapReduce and Pregel that typically have all-to-all communication structures. However, for the applications with different communication structures like multi-tiered applications, the hose model based abstractions can cause low resource utilization. To address this issue, CloudMirror [8] drives the virtual network topology based on the application communication structures. By accurately capturing real communication patterns among VMs, it can effectively reduce the bandwidth reserved for the virtual cluster and improve rejection rate of virtual cluster requests from tenants. To derive

the network abstractions in CloudMirror, it is required to know the type of applications, their communication structures and to decide different functions for different VMs among tiers. In our work, we focus on showing the advantage of incorporating bandwidth demand uncertainty information in the network abstractions rather than considering application communication structures. Thus, we assume unknown application communication structures or simple all-to-all communication structures and choose host model based abstractions.

To achieve bandwidth guaranteed virtual to physical mapping for different virtual network abstractions, the previous works also propose various VM allocation algorithms and the mechanisms to enforce the bandwidth guarantee. The work [20] also considers the dynamic scaling of virtual cluster abstractions in terms of the number of VMs and bandwidth requirement in Oktopus and proposes efficient VM alocation algorithms to support it. However, they require exact and deterministic bandwidth demand information for the virtual network abstractions, which is difficult to obtain due to traffic burstiness and demand uncertainty. These approaches have to specify over-provisioned bandwidth for virtual networks to deal with the stochastic traffic. While ensuring the network performance isolation among different tenants, these approaches enforce bandwidth guarantee through static provision of link bandwidth. Thus, they are not work-conserving, which leads to inefficient utilization of the network bandwidth.

Gatekeeper [14] and EyeQ [15] are two work-conserving solutions for bandwidth guarantee. They enforce the minimum bandwidth guarantees with using the horse model as virtual network abstractions. Based on the assumption that the core of the network fabric is congestion-free and the bottlenecks occur at the endpoint links, they ensure minimum bandwidth guarantee while achieving work-conservation through rate-based congestion control at end-points. Another work-conserving mechanism for bandwidth guarantee enforcement, named ElasticSwitch [16], is proposed recently. ElasticSwitch transforms the bandwidth guarantee specified by the hose model into pairwise VM-to-VM rate-limits. It achieves work conservation by dynamically increasing the rate-limits when the network is not congested. The problem of these approaches is that their admission control and VM placement of virtual clusters do not take into account the traffic burstiness, because they are based on the hose model with deterministic bandwidth requirement. This may lead to inefficient VM placement with regard to network utilization. Even with the dynamic rate allocation at runtime, it would limit the room for improvement of network utilization.

Another type of network sharing solutions, such as Seawall [4], Netshare [18] and FairCloud [17], share the network bandwidth proportionally based on weights. Seawall [4] proposes a VM-level congestion control approach to ensure the share of bandwidth obtained by per source VM in each congested network link is proportional to its weight. The congestion control approach adjusts rates with weighted additive rate increase and multiplicative rate decrease functions. Netshare [18] proposes a hierarchical weighted max-min fair sharing mechanism which allocates the bandwidth to services according to their weights with a central controller. For each service, Netshare allocates its bandwidth equally to its TCP connections. Faircloud [17] describes a comprehensive set of properties for cloud network sharing, including proportionality, minimum bandwidth guarantee, work conservation, etc. It proposes three allocation policies to navigate the tradeoff space among these properties. A bandwidth allocation algorithm, Falloc [21] is proposed, that uses the cooperative game model to solve the datacenter bandwidth allocation problem with achieving both minimum bandwidth guarantee and proportional bandwidth share for the residual bandwidth. Seawall and Netshare cannot provide minimum bandwidth guarantees, because the proportion of bandwidth allocation on the links depends on other source VMs or tenants and can be arbitrarily reduced. Although a policy PS-P in Faircloud can provide minimum bandwidth guarantee, it requires extra hardware support of switches and tree-based datacenter topologies.

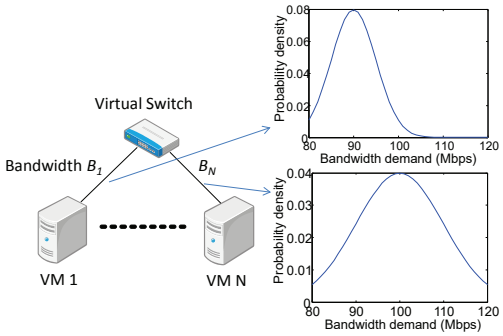


Fig. 1: Virtual cluster model with stochastic bandwidth demand.

2.2 Statistical Traffic Engineering

A statistical traffic engineering framework [11] has been proposed for optimizing bandwidth provisioning and route selection under demand uncertainty. It uses probabilistic distribution of demands as inputs for off-line optimization with respect to the paths serving end-to-end traffic demand and the amount of provisioned bandwidth on these paths. The optimization objective is to maximize the revenue obtained from serving demands. The numerical results show that demand uncertainty has significant impacts on mean revenue and utilization. In this paper we integrate distributional information of bandwidth demands into virtual network abstraction for bandwidth guarantee, in order to optimize the admission of virtual clusters and VM placement under demand uncertainty with the goal to improve the network utilization.

3 STOCHASTIC NETWORK ABSTRACTION AND FRAMEWORK

In this section we introduce a new virtual cluster network abstraction with stochastic bandwidth demand to account for demand uncertainty, and propose a stochastic framework for virtual cluster allocation with probabilistic bandwidth guarantee.

3.1 Stochastic Virtual Cluster Model

Highly dynamic bandwidth usage of cloud applications [9], [10] indicates the need for a new networking abstraction that can express the demand uncertainty of application networking requirement. To this end, we propose a novel network abstraction called *Stochastic Virtual Cluster model* (SVC) that captures the bandwidth demand uncertainty of cloud applications.

The SVC abstraction (shown in Fig. 1) consists of a virtual cluster of N nodes VM 1, ..., VM N, connected to a switch, via links of bandwidth $B_1, \ldots, B_N$ respectively, similar to the virtual cluster model in [6]. However, there are two key differences. First, the bandwidth for each link is a random variable, instead of a constant value in the previous work [6]. This not only avoids the need for reliable bandwidth estimate which is impossible for

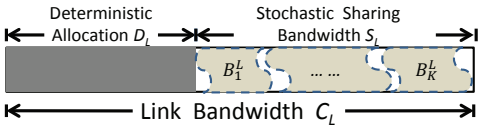


Fig. 2: View of Bandwidth Allocation on link L.

highly dynamic network traffic, but also improves the utilization of datacenter resources and the performance of cloud applications by efficiently capturing the uncertainty of the future bandwidth usage of applications, as shown in our simulations. Second, the links can have heterogeneous bandwidth, i.e., the bandwidth of different links have different probability distributions, instead of homogeneous constant bandwidth demand for all VMs [6]. This provides more flexibility to characterize the diverse bandwidth needs of tenants' applications.

The first step to derive a SVC abstraction for the customers' cloud applications is to estimate the probability distributions of bandwidth demands of VMs in their jobs. Similar to previous works [5], [6], [7], cloud customers can conduct the profiling runs before the production runs of their applications, during which the traffic usage information of VMs is sampled at all VMs to log all communications between VMs with network tools such as tcpdump, or sFlow and NetFlow that are already supported by software switches like Open vSwitch (OVS). With the bandwidth usage samples during the profiling runs, the probability distribution of bandwidth demands can be estimated for every VM. Then, the customers can have the final SVC model description for the production runs of their cloud applications, which specifies the number of VMs N and the distribution information of $B_1, \ldots, B_N$. Note that the profiling step adds additional overhead but the overhead can be amortized and drastically reduced when the same type of cloud applications with the same input size is repeatedly run many times.

3.2 Probabilistic Bandwidth Guarantee for Stochastic Bandwidth Demand

For a virtual cluster with deterministic bandwidth, the amount of reserved bandwidth on any physical links only depends on the placement of its VMs. The bandwidth is guaranteed via ensuring sufficient bandwidth provision of the physical links that connect the virtual cluster's VMs and enforcing bandwidth reservation by rate limiting at VMs or switches. For SVC model, however, the problem of bandwidth guarantee is different due to the stochastic bandwidth requirement. In this paper we introduce *probabilistic bandwidth guarantee* for SVC.

We first characterize bandwidth occupation of K allocated virtual clusters with stochastic bandwidth demands on a physical link L. Let C_L be the bandwidth capacity of link L. We assume that the virtual clusters with deterministic and stochastic bandwidth requirements can co-exist. A deterministic portion of link bandwidth is reserved for the deterministic virtual clusters, and the residual bandwidth is shared among the stochastic virtual clusters. Denote the total amount of the reserved bandwidth for deterministic virtual clusters on link L by D_L. Then, as shown in Fig. 2, the residual bandwidth $S_L = C_L - D_L$ is shared among K stochastic virtual clusters, called *stochastic sharing bandwidth*.

Let $B_1^L, \ldots, B_K^L$ be the random bandwidth demands of K virtual clusters on link L, as shown in Fig. 2. The probabilistic bandwidth guarantee for these K virtual clusters means that link L can satisfy their bandwidth demands with a high probability $1 - \epsilon$, i.e.,

$$\Pr(\sum_i B_i^L > S_L) < \epsilon. \quad (1)$$

This inequality describes that the bandwidth outage on link L is only allowed to happen with a small probability ϵ. For a specific virtual cluster with bandwidth B_i^L, the inequality (1) indicates $\Pr(B_i^L < S_L - \sum_{j \neq i} B_j^L) > 1 - \epsilon$, i.e., the bandwidth of this virtual cluster is guaranteed with a high probability. The parameter ϵ is indeed a risk factor for link bandwidth shortage with regard to the tenants' demands. It can be determined by the cloud provider as a part of a service level agreement.

3.3 Framework for Virtual Cluster Allocation

The above stochastic virtual cluster abstraction with probabilistic bandwidth guarantee defines a framework for virtual cluster admission and VM placement that takes into account the demand uncertainty.

In this framework, provided the virtual cluster request from a tenant with the probability distribution of bandwidth demand, the admission control mechanism with an efficient VM placement algorithm determines if the virtual cluster can be deployed on the physical network without violating the probabilistic bandwidth guarantees for all the existing tenants. A network manager in the data center, upon receiving a tenant request, performs admission control and VM placement, with physical links satisfying the bandwidth requirements in terms of the probabilistic constraint (1).

The SVC based framework aims to find proper allocation of virtual clusters with the consideration of bandwidth demand uncertainty. To solve the VM allocation problem in an online fashion, the network manager maintains the up-to-date status of the datacenter network, including (1) the datacenter network topology; (2) the empty slots in each physical machine (PM, for short) ; (3) the stochastic sharing bandwidth S_L on each physical link L, calculated from counting the allocations of deterministic virtual clusters running in the datacenter; (4) the probability distribution of bandwidth demand of any existing SVC allocations on each link. Our design of VM allocation algorithm focuses on tree topology. Such a topology is hierarchical, in which machines are grouped into racks and the Top-of-Rack (ToR) switches are in turn connected to higher level switches. In today's datacenters, the network can have more complex topologies like multi-rooted trees and fat-trees. Our algorithms can be applied to these tree-like topologies because they are also hierarchical and recursively composed of sub-trees at each level. Because these networks usually use load balancing techniques to spread traffic across multiple equal cost paths, as discussed in [7], our algorithms can follow the same strategy to be applied here, by regarding multiple links from each PM or switch that are used in multiple equal cost paths to the same destination as a single aggregation link. The bandwidth reservation on an aggregation link can be enforced by having equal reservation split among the multiple physical links in the aggregate.

An SVC request from tenants specifies the number of VMs N, and the probability distribution of each VM's bandwidth demand, denoted by $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N$. For instance, when the bandwidth demand follows normal distribution, the probability distribution is given by its mean and variance. After receiving the request, the network manager tries to find N empty VM slots to place requested VMs such that each physical link can still satisfy the constraint (1) for all stochastic bandwidth demands it carries. Note that the

tenants can also specify constant bandwidth demand for each VM in SVC as in the traditional virtual cluster proposed in [6]. The deterministic and stochastic bandwidth allocation can co-exist in the datacenters in our framework.

4 VIRTUAL MACHINE ALLOCATION FOR HOMOGENEOUS BANDWIDTH DEMAND

In this section, we address the VM allocation problem for SVC model with homogeneous bandwidth demand, which means that the bandwidth demand of every VM follows the same probability distribution, i.e., $\mathcal{P}_i = \mathcal{P}$ $(1 \leq i \leq N)$. The VM allocation problem has been shown to be NP-hard for deterministic virtual cluster model [6], [19]. To solve the problem, we first derive the condition of a valid VM allocation for stochastic bandwidth demands, and based on that we propose dynamic programming based VM allocation algorithms with the goal to achieve both good locality of VMs and less bandwidth occupancy on links.

4.1 Determining valid allocation

We begin with characterizing the stochastic bandwidth demand of an SVC allocation on a link. Under homogeneous SVC model, the bandwidth demands of N VMs are independent and identically distributed (i.i.d.) random variables following distribution $\mathcal{P}$. Considering a link L on the tree topology, removing L from the tree results in two disconnected network components. Suppose one component contains m allocated VMs, and accordingly the other one contains $N - m$ VMs. The aggregate bandwidth demand of m VMs has distribution $\mathcal{B}(m) = \sum_{i=1}^{m} B_i$ where B_i is a random variable of distribution $\mathcal{P}$, and the other component's bandwidth demand has distribution $\mathcal{B}(N-m)$. Because virtual machines in one component cannot send data at a rate larger than the other component's total receiving rate, the total bandwidth demand of such SVC for link L, denoted by $B_r^L(m)$, is $\min(\mathcal{B}(m), \mathcal{B}(N-m))$, which can be computed with given distribution function $\mathcal{P}$ in the SVC request.

Assume that link L currently serves the SVC requests $r_1, \ldots, r_K$ with stochastic sharing bandwidth S_L (as shown in Fig. 2), and the constraint (1) is satisfied. Each SVC r_i has bandwidth demand $B_i^L = \min(\mathcal{B}(m_i), \mathcal{B}(N_i - m_i))$ on link L, where N_i is VM number requested in r_i and link L divides its VMs into two groups of m_i VMs and $N_i - m_i$ VMs respectively. For a new arrival request r_{K+1}, a valid allocation needs to ensure the adding of demand $B_{K+1}^L(m)$ to link L does not violate the constraint (1), i.e., still $\Pr(\sum_{i=1}^{K+1} B_i^L > S_L) < \epsilon$.

Since $B_1^L, \ldots, B_{K+1}^L$ are bandwidth demands for link L under the VM allocations for each different SVC, they are assumed to be independent random variables. Accordingly, we use the normal distribution to approximate the distribution of $B^L = \sum_{i=1}^{K+1} B_i^L$ according to the central limit theorem. Denote the mean and variance of B_i^L by $\mu_{i,L}$ and $\sigma_{i,L}^2$. Then, we have $B^L \sim \mathcal{N}(\sum_{i=1}^{K+1} \mu_{i,L}, \sum_{i=1}^{K+1} \sigma_{i,L}^2)$. Since $\frac{B^L - E(B^L)}{\sqrt{(Var(B^L))}} \sim \mathcal{N}(0,1)$, to satisfy the constraint (1), we can easily derive the following condition

$$\frac{S_L - \sum_{i=1}^{K+1} \mu_{i,L}}{\sqrt{\sum_{i=1}^{K+1} \sigma_{i,L}^2}} > \Phi^{-1}(1-\epsilon), \tag{2}$$

where $\Phi^{-1}(\cdot)$ is the inverse function of $\Phi(\cdot)$.

The inequality (2) gives us the sufficient condition for a valid allocation to an SVC request. Besides, a valid allocation should only allocate VMs onto the empty slots on PMs. Accordingly, given a VM allocation solution for SVC request r that allocates m VMs and $N - m$ VMs in two network components divided by any link L, we check (i) whether there are no less than m and $N - m$ empty VM slots in two components respectively, and (ii) whether the condition (2) still holds with the mean and variance of distribution $\min(\mathcal{B}(m), \mathcal{B}(N-m))$. An allocation solution for SVC is valid only if the above two conditions are checked successfully for any physical links in the network. Note that if the new arrival request r_{K+1} is a deterministic virtual cluster request with VM bandwidth B, i.e., $\mu_{K+1,L} = B\min(m, N-m)$ and $\sigma_{K+1,L}^2 = 0$, we can still use the condition (2), which actually becomes to verify whether the constraint (1) can still be met for serving the existing stochastic request $r_1, \ldots, r_K$ under the new stochastic bandwidth $S'_L = S_L - u_{K+1,L}$. If there are only deterministic bandwidth demands for the link, we only need to verify the sum of bandwidth reservations is less than the link capacity.

VM bandwidth of normal distribution: A virtual cluster uses a physical link for the communication between its VM pairs split by this link. When the traffic between these VM pairs comes from many independent individual flows, the aggregated traffic demand on the link can be approximated by the normal distribution. It has been extensively observed that the aggregated traffic demand between network nodes follows the normal distribution [12], [13]. Thus, here we consider SVC and its valid allocation in a special case that the bandwidth demand B of VMs follows a normal distribution $\mathcal{N}(\mu, \sigma^2)$ as in [22], [23], with mean $\mu = E(B)$ and variance $\sigma = Var(B)$. Then, an SVC request can be represented by $<N, (\mu_1, \sigma_1), (\mu_2, \sigma_2), \ldots, (\mu_N, \sigma_N)>$, in which the bandwidth demand for each VM i follows normal distribution $\mathcal{N}(\mu_i, \sigma_i^2)$.

Homogeneous SVC is represented by $r = <N, \mu, \sigma>$, where each VM's bandwidth demand follows the same normal distribution $\mathcal{N}(\mu, \sigma^2)$. The aggregate bandwidth demand of m VMs $\mathcal{B}(m)$, has distribution $\mathcal{N}(m\mu, m\sigma^2)$. The variable $B_r^L(m) = \min(\mathcal{B}(m), \mathcal{B}(N-m))$ is the min of two normal variables. Based on [24] that provides the exact distribution of the min of two normal variables, we can easily derive the following results:

Lemma 1. For two independent normal variables $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$, the mean and variance of $X = \min(X_1, X_2)$ are

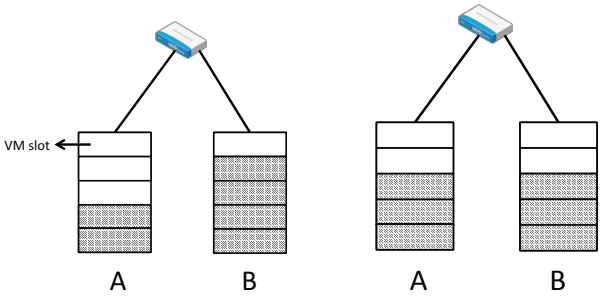
$$E(X) = \mu_1\Phi(\alpha) + \mu_2\Phi(-\alpha) - \theta\phi(\alpha) \tag{3}$$

$$Var(X) = (\sigma_1^2+\mu_1^2)\Phi(\alpha)+(\sigma_2^2+\mu_2^2)\Phi(-\alpha)-(\mu_1+\mu_2)\theta\phi(\alpha)-(E(X))^2 \tag{4}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are the probability density function (pdf) and the cumulative distribution function (cdf) of the standard normal distribution, respectively, and $\theta = \sqrt{\sigma_1^2 + \sigma_2^2}$ and $\alpha = \frac{\mu_2 - \mu_1}{\theta}$.

By Lemma 1, we can compute the mean and variance of $\min(\mathcal{B}(m), \mathcal{B}(N-m))$, i.e., $\mu_{r,L}$ and $\sigma_{r,L}^2$, respectively. Accordingly, we can check the condition (2).

It is worth to note that the SVC model, its allocation framework, and the following VM allocation algorithms do not depend on any specific probability distributions. The tenants can choose different types of probability distributions to characterize uncertain VM bandwidth demands. Lemma 1 is for normal distribution. In the cases other than normal distribution, the probability distributions of the aggregate bandwidth of VMs $\mathcal{B}(m)$ and $\mathcal{B}(N-m)$ in SVC as well as $\min(\mathcal{B}(m), \mathcal{B}(N-m))$ need to be computed according to the chosen distribution.



(a) 2 VMs in A, 4 VMs in B (b) 3 VMs in A, 3 VMs in B

Fig. 3: Two valid allocations with different bandwidth occupancys.

4.2 VM allocation algorithm

According to the discussion above, VM allocation in the tree topology is to allocate an SVC to a subtree, in which there are enough empty VM slots and any link L can satisfy the bandwidth requirement of VMs placed in the subtree connected by L to the upper level.

The previous work TIVC [7] proposes a dynamic programming based searching algorithm, referred to as TIVC searching algorithm, to find the lowest subtree for the allocation of the deterministic virtual cluster abstraction. Searching the lowest possible subtree is for the most localized allocation of VMs such that the bandwidth of the links in the upper levels of the tree is conserved and the ability to accommodate future tenant requests is maximized. However, this algorithm cannot be directly used to solve our VM allocation problem since it is based on deterministic bandwidth demands, and also it cannot achieve optimal bandwidth occupancy that is addressed by our allocation algorithm.

For TIVC searching algorithm, multiple possible valid allocations may exist for the same subtree. But because the algorithm makes no distinction between them, it may result in suboptimal allocation in term of bandwidth occupancy, i,e, the amount of bandwidth reserved on links. To illustrate this problem, Fig. 3 gives an example, with a simple tree topology T in which a switch connects two PMs A and B each having 5 VM slots and each link has bandwidth capacity 50. Considering a deterministic virtual cluster request $<N = 6,\ B = 10>$ that requires 6 VMs each with bandwidth 10, we can see that the tree T is indeed the lowest subtree the algorithm finds. The allocation in Fig. 3(a) with 2 VMs in A and 4 VMs in B, and the allocation in Fig. 3(b) with 3 VMs in A and 3 VMs in B are both valid. The reserved bandwidth on two links in Fig. 3(a) is $10 \times \min(2, 4) = 20$, lower than 30 in Fig. 3(b), which means that the former has a lower bandwidth occupancy. This example indicates that the VM allocation solution returned by TIVC algorithm [7] can be sub-optimal on the metric of bandwidth occupancy. The reason of that is formally shown in our algorithm description later.

In this paper we propose the allocation algorithm for SVC with the goal of finding the optimal VM allocation that fits in the lowest-level subtree while minimizing the bandwidth occupancy of the links in the subtree.

4.2.1 Bandwidth occupancy

We first quantify the bandwidth occupancy of a link. Without stochastic bandwidth demands on link L, its bandwidth occupancy can be easily measured as the ratio of $\frac{D_L}{C_L}$ where D_L is the amount of deterministic reserved bandwidth and C_L is the total bandwidth capacity of link L. However, for the bandwidth occupancy with stochastic demands, the answer is not so obvious.

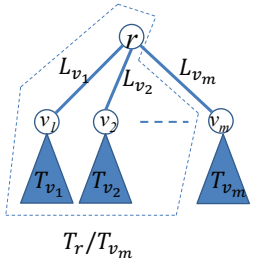


Fig. 4: The diagram for showing the optimal substructure with Lemma 2.

To characterize the bandwidth occupancy of a link with stochastic demands, we introduce the concept of *effective amount* of stochastic bandwidth demand. Consider link L shown in Fig. 2 that meets the constraint (1). According to Inequality (2), we have

$$S_L > \sum_{i=1}^{K} \mu_{i,L} + c\sqrt{\sum_{i=1}^{K} \sigma_{i,L}^2} = \sum_{i=1}^{K} \left(\mu_{i,L} + c\frac{\sigma_{i,L}^2}{\sqrt{\sum_{i=1}^{K} \sigma_{i,L}^2}} \right) \quad (5)$$

where $c = \Phi^{-1}(1-\epsilon)$. With the above transformation of the right side of Inequality (5), we can regard $\mu_{i,L} + c\frac{\sigma_{i,L}^2}{\sqrt{\sum_{i=1}^{K} \sigma_{i,L}^2}}$ as the *effective amount* of bandwidth reserved for stochastic demand B_i^L, denoted by E_i^L $(1 \le i \le K)$. As we can see, it depends on the other demands served by the links since they statistically share the bandwidth, rather than being individually reserved in previous deterministic virtual cluster models. Then, we measure the bandwidth occupancy ratio of link L, denoted by O_L, as follows:

$$O_L = \frac{1}{C_L}\left(D_L + \sum_{i=1}^{K} E_i^L \right) \quad (6)$$

Note that $S_L > \sum_{i=1}^{K} E_i^L$ is equivalent to $O_L < \frac{1}{C_L}(D_L + S_L) = 1$. Thus, the sufficient condition (2) for a valid allocation can also expressed as $O_L < 1$ for any link L.

4.2.2 Minimize the maximum of the bandwidth occupancy ratios

The example in Fig. 3 shows a special case that two links always have the same bandwidth occupancy ratios given only one request in the network, so we can minimize their bandwidth occupancy ratio simultaneously. However, in general network topologies, the bandwidth occupancy ratios vary among different links and have dependencies due to shared bandwidth demands, so it is not feasible to simultaneously to minimize the bandwidth occupancy ratio for each link. On the other hand, since our SVC model guarantees the bandwidth in a probabilistic way, link congestion can occur when $\sum_i B_i^L > S_L$ in the constraint (1). The link with the maximum of bandwidth occupancy ratios is the most likely congested link in the datacenter. Thus, we expect to reduce the possibility of congestion by the optimization to minimize the maximum bandwidth occupancy ratio. Therefore, our goal is to find the valid allocation which minimizes the maximum of the bandwidth occupancy ratios of the links in the subtree.

We now show that this min-max problem has optimal substructure. Assume that a tree T_r rooted at vertex r has m children $v_1, \ldots, v_m$, as shown in Fig. 4. Each link between v_i and r is denoted by L_{v_i}, also called the uplink of v_i. Let T_{v_i} be the subtree rooted at v_i.

Definition 1 (Allocable VM set). For an SVC request of N VMs, the set of all possible numbers of VMs out of the N VMs that can be allocated in the subtree T_v rooted at v, with satisfying the bandwidth constraint of any link in T_v and also the uplink of v, is called *allocable VM set* of T_v (or v).

Suppose we have n VMs to be allocated to T_r, and let A^* be the optimal VM allocation that minimizes $\max_{L\in T_r}(O_L)$ where L is a link of T_r. Suppose that the allocation A^* assigns $n^*_{v_m}$ number of VMs to T_{v_m}, then $(n - n^*_{v_m})$ VMs are assigned to $T_r \setminus T_{v_m}$ which is a tree formed by removing T_{v_m} from T_r. Given that n_v VMs out of the total n VMs are allocated to T_v, $\max_{L\in T_v}(O_L)$ actually only varies with the placement of n_v VMs in T_v, regardless of the placement of $n - n_v$ VMs in the rest of the network $T_r \setminus T_v$. Thus, we assume $Opt(T_v, n_v)$ be the minimum value of $\max_{L\in T_v}(O_L)$ among all possible allocations of n_v VMs in T_v. Then, we have the following lemma:

Lemma 2.

$$Opt(T_r, n) = \max\left\{Opt(T_{v_m}, n^*_{v_m}), Opt(T_r \setminus T_{v_m}, n - n^*_{v_m}), O^*_{L_{v_m}}\right\} \tag{7}$$

where $O^*_{L_{v_m}}$ is the bandwidth occupancy ratio of link L_{v_m} under the allocation A^*.

Proof: Under the allocation A^*, let the link that has the optimal bandwidth occupancy ratio of $Opt(T_r, n)$ be L^*, i.e., $O_{L^*} = Opt(T_r, n)$. Denote the maximum of the bandwidth occupancy ratio of links in T_{v_m} and $T_r \setminus T_{v_m}$ under A^* by $O^*(T_{v_m})$ and $O^*(T_r \setminus T_{v_m})$, respectively. Obviously, we have

$$O_{L^*} \geq O^*(T_{v_m}) \geq Opt(T_{v_m}, n^*_{v_m}) \tag{8}$$

$$O_{L^*} \geq O^*(T_r \setminus T_{v_m}) \geq Opt(T_r \setminus T_{v_m}, n - n^*_{v_m}) \tag{9}$$

$$O_{L^*} \geq O^*_{L_{v_m}} \tag{10}$$

Then, we prove the lemma by three cases of L^*:

1. If $L^* = L_{v_m}$, we have $O^*_{L_{v_m}} = O_{L^*} \geq O^*(T_{v_m}) \geq Opt(T_{v_m}, n^*_{v_m})$. Similarly, we have $O_{L_{v_m}} \geq Opt(T_r \setminus T_{v_m}, n - n^*_{v_m})$. Thus, (7) holds.

2. If $L^* \in T_{v_m}$, we have $O_{L^*} = O^*(T_{v_m}) \geq Opt(T_{v_m}, n^*_{v_m})$. If $O_{L^*} > Opt(T_{v_m}, n^*_{v_m})$, by allocating $n^*_{v_m} VMs$ to T_{v_m} according to the allocation that achieves $Opt(T_{v_m}, n^*_{v_m})$, we can obtain smaller $\max_{L\in T_r}(O_L)$, which is a contradiction to the optimality of O_{L^*}. Thus, we must have $O_{L^*} = Opt(T_{v_m}, n^*_{v_m})$. Then, according to (9) and (10), we can also tell that $Opt(T_{v_m}, n^*_{v_m})$ is the maximum of $\{Opt(T_{v_m}, n^*_{v_m}), Opt(T_r \setminus T_{v_m}, n - n^*_{v_m}), O^*_{L_{v_m}}\}$. Thus, (7) holds.

3. If $L^* \in T_r \setminus T_{v_m}$, we can prove (7) as in the case of $L^* \in T_{v_m}$. □

Lemma 2 shows the optimal substructure of the problem. Accordingly, given the optimal values of the child subtrees, the optimal value $Opt(T_r, n)$ can be found by searching optimal $n^*_{v_m}$. So we can give the dynamic programming recursive formula to compute the optimal value as follows:

$$Opt(T_r, n) = \min_{\substack{x\in M_{v_m} \\ n-x\in M_{-v_m}}} \max\left\{Opt(T_{v_m}, x), Opt(T_r \setminus T_{v_m}, n - x), O_{L_{v_m}}(n, x)\right\} \tag{11}$$

Here x is the number of VMs allocated into T_{v_m}. M_{v_m} and M_{-v_m} are the allocable VM sets of T_{vm} and $T_r \setminus T_{v_m}$, respectively, with considering the bandwidth constraint of L_{v_m}. $O_{L_{v_m}}(n, x)$ denotes the bandwidth occupancy ratio of link L_{v_m} given x VMs in T_{v_m} and $n - x$ VMs in $T_r \setminus T_{v_m}$. $O_{L_{v_m}}(n, x)$ is a function of n and x, which can be calculated with (3), (4) and (6). Both $O_{L_{v_m}}(n, 0)$ and $O_{L_{v_m}}(n, n)$ are equal to the initial bandwidth occupancy ratio based on existing SVC demands on the link.

If the subtree T_v has only one child subtree T_{v_1}, which means $T_v \setminus T_{v_1}$ is the root vertex, then

$$Opt(T_v, x) = \max\left\{Opt(T_{v_1}, x), O_{L_{v_1}}(n, x)\right\} \tag{12}$$

With (11) and (12), we can use dynamic programming to find the optimal allocation in a given tree.

4.2.3 Algorithm description

Now we present our allocation algorithm. The algorithm traverses the topology tree starting at the leaves (PMs at level 0) and determines if all N VMs in a request can fit. During the traverse, for any visited vertex v, the algorithm records its allocable VM set, which is calculated by reusing the recorded allocable VM sets of v's children with consideration of the optimality of bandwidth occupancy. Since searching the allocable lowest-level subtree and optimizing $\max_{L\in T}(O_L)$ are both dynamic programming procedures, we propose an efficient algorithm which combines the optimization into the searching within only one tree traversal. Algorithm 1 shows our VM allocation algorithm in pseudo code.

In Algorithm 1, $T_v[i]$ denotes the tree that consists of vertex v as the root and v's first i child subtrees. Take Fig. 4 for instance, $T_r[m-1] = T_r \setminus T_{v_m}$. We define $T_v[0] = \{v\}$ and $T_v[i] = T_v[i-1] \bigoplus T_{v_i}$ where "$\bigoplus$" is to connect the child subtree T_{v_i} to $T_v[i-1]$ via link L_{v_i}. $T_v = T_v[m]$ where m is the number of v's children. $S_v[i]$ denotes the set that contains the numbers of VMs that could be accommodated $T_v[i]$, without considering the uplink bandwidth constraint of v. M_v denotes the allocable VM set of v. We have dynamic programming step for minimizing the bandwidth occupancy ratio in lines 19∼25. The lines 20 and 22 are corresponding to (11) and (12). Compared with TIVC algorithm [7], a key difference exists when recording the allocation in the traversed subtrees. That is, for each possible value $e+h$ in $S_v[i]$, our algorithm records in $D_v[i, h]$ the number of VMs assigned to the i-th child of v that minimizes the maximum bandwidth occupancy ratio of links in $T_v[i]$, considering that there may be multiple combinations of e and h achieving same sum $e+h$. TIVC algorithm, however, does not make such an optimal choice for bandwidth occupancy, thus it may return suboptimal allocation. Then, as in TIVC algorithm, if N can be allocated according to M_v, Alloc() is called recursively according to $D_v[i, x]$ while recording the statistical information of bandwidth demand under the allocation for request r on each link, and eventually obtain the number of VMs per PM. The time complexity of Algorithm 1 is $O(|V|\Delta N^2)$ where $|V|$ is the number of vertices in T and Δ is the maximum number of children of any nodes.

5 VM ALLOCATION FOR HETEROGENEOUS BANDWIDTH DEMAND

In this section, we address the VM allocation problem for heterogeneous SVC. For normal distribution, the SVC request can be represented by $r = \langle N, (\mu_1, \sigma_1), (\mu_2, \sigma_2), \ldots, (\mu_N, \sigma_N)\rangle$, which means the bandwidth demands of VMs B_i $(1 \leq i \leq N)$ may have different probability distributions $\mathcal{N}(\mu_i, \sigma_i^2)$. The problem is also NP-hard in deterministic case [19].

5.1 Determining valid allocation

Under an allocation for the heterogeneous SVC model, a link L on the tree divides N VMs into two VM sets $\mathbb{V}_{L1}$ and $\mathbb{V}_{L2}$. Accordingly, their bandwidth demands $\{B_i | 1 \leq i \leq N\}$ are divided into two sets, denoted by $\mathbb{B}_{L1}$ and $\mathbb{B}_{L2}$. The aggregate bandwidth demand for $\mathbb{V}_{Li}$, denoted by $\mathcal{B}(\mathbb{B}_{Li})$, follows the normal

Algorithm 1: VM Allocation Algorithm

Input: Datacenter tree topology T, SVC request $r = \langle N, \mu, \sigma \rangle$, bandwidth reservation information of each link including the mean and variance of each SVC demand on it

for *level* $l \leftarrow 0$ *to Height(T)* **do**
  for *each subtree* T_v *rooted at vertex* v *at level* l **do**
    $m \leftarrow$ the number of v's children;
    if $l = 0$ **then** // leaf v is a PM
      $S_v[0] \leftarrow \{0, 1, \ldots, C_v\}$; // C_v is the number of empty VM slots of v
      for *each value* h *in* $S_v[0]$ **do**
        $Opt[T_v, h] = 0$; // No link usage between VMs in the same machine
    else
      $S_v[0] \leftarrow \{0\}$;
    $T_v[0] \leftarrow v$;
    for i *from 1 to* m **do**
      $S_v[i] \leftarrow \{0\}$;
      $T_v[i] \leftarrow T_v[i-1] \bigoplus T_{v_i}$;
      for *each value* e *in* v*'s* i*-th child* v_i*'s allocable VM set* M_{v_i} **do**
        for *each value* h *in* $S_v[i-1]$ **do**
          if $e + h$ *is not in* $S_v[i]$ **then**
            $min_v[i, e+h] \leftarrow \infty$;
            $S_v[i] \leftarrow S_v[i] \cup \{e+h\}$;
          if $i = 1$ **then**
            $Opt[T_v[i], e+h] \leftarrow \max\left\{Opt[T_{v_i}, e+h], O_{L_{v_i}}(N, e+h)\right\}$
          else
            $Opt[T_v[i], e+h] \leftarrow \max\left\{Opt[T_v[i-1], h], Opt[T_{v_i}, e], O_{L_{v_i}}(N, e)\right\}$
          if $Opt[T_v[i], e+h] < min_v[i, e+h]$ **then**
            $D_v[i, e+h] \leftarrow e$;
            $min_v[i, e+h] \leftarrow Opt[T_v[i], e+h]$;
    $M_v \leftarrow \emptyset$;
    for *each value* h *in* $S_v[m]$ **do**
      if $O_{L_v}(N, h) < 1$ **then** // L_v is the uplink of v
        $M_v = M_v \cup \{h\}$;
        $Opt[T_v, h] \leftarrow min_v[m, h]$;
    if $N \in M_v$ **then**
      Alloc (r, v, N);
      return true;
return false;
Procedure Alloc(r, v, x)
  if v *is a machine* **then**
    allocate x VMs in v;
  else
    for v*'s child* i *from* m *to 1* **do**
      Alloc$(r, v_i, D_v[i, x])$;
      record bandwidth occupancy for r on v's i-th link;
      $x = x - D_v[i, x]$;

distribution $\mathcal{N}(\sum_{B_i \in \mathbb{B}_{Li}} u_i, \sum_{B_i \in \mathbb{B}_{Li}} \sigma_i^2)$. Similar to the homogeneous case, the bandwidth demand for request r on link L, denoted by $B_r^L(\mathbb{B}_{L1}, \mathbb{B}_{L2})$, is $\min(\mathcal{B}(\mathbb{B}_{L1}), \mathcal{B}(\mathbb{B}_{L2}))$. Then, we still use Lemma 1 and Inequality (2) to check the validity of an allocation.

5.2 VM Allocation Algorithm

The VM allocation algorithm for the homogeneous virtual cluster model cannot be used in the heterogeneous case. For the homogeneous model, all VMs are the same (i.e., with the same bandwidth requirement), and their allocation does not need to distinguish different VMs. As a result, the allocation algorithm for homogeneous model only need to decide the number of VMs in each subtree. However, in the heterogeneous case, every VM can have different bandwidth demand, and thus the allocation needs to decide which VM is allocated to which subtree. The allocation algorithms in previous works [6], [7] regard VMs are homogeneous and thus they only need to decide the number of VMs in every subtree. The allocation algorithm for the heterogeneous model has to consider which subset of VMs is allocated to which subtree. It is worth to note that CloudMirror's algorithm [8] can be extended to deal with the heterogeneous model by regarding each VM as a tier in the TAG model (i.e., one tier has one VM). Since in our paper we do not assume any prior known communication structure of cloud applications, here we propose a dynamic programming based search algorithm for the heterogeneous case.

Dynamic programming based allocation algorithm. We can extend the dynamic programming (DP) approach in Algorithm 1 to find the optimal allocation for the heterogeneous model, with maintaining the set of all possible VM subsets that can be allocated in each subtree. Accordingly, in the recursive formula (11), the allocable VM sets M_{v_m} and M_{-v_m} are redefined as the sets consisting of the VM subsets that can be allocated in the corresponding subtree T_{v_m} and $T_r \setminus T_{v_m}$. However, this approach is much more costly for the heterogeneous model to find the optimal allocation than for the homogeneous model. In the homogeneous case, the number of possible VM subsets that can be allocated to any subtree is at most $N + 1$, since the VMs are homogeneous and indistinguishable and the only variable is the size of the VM subset. In the heterogeneous case, however, the number of possible VM subsets that can be allocated to any subtree, i.e., the size of any allocable VM set, is at most $O(2^N)$. Therefore, the algorithm has exponential time complexity $O(|V|\Delta 2^N)$, which can be applied for small N but is infeasible for large N.

Heuristic allocation algorithm. For the above algorithm, its time complexity depends on the sizes of the allocable VM sets, which indicates that we can reduce the time complexity for large N by limiting the size of each allocable VM set with some heuristic. Therefore, we propose a heuristic algorithm, which identifies an allocable VM set with polynomial size in N for each subtree during tree traversal.

Our algorithm is derived from a simple First Fit algorithm FF. In FF, VMs are sorted by their bandwidth demands and then placed sequentially in the first subtree having sufficient bandwidth and empty VM slots. In the case of the deterministic bandwidth demands, let $S_v = (v_1, v_2 \ldots, v_n)$ be a sequence of n VMs in ascending order of their bandwidth demands. To allocate S_v to the tree T_r rooted at r, T_r's PMs are visited from left to right, and the algorithm greedily and sequentially places as many VMs as possible into each PM in the order of S_v. The algorithm finds the longest sequence of VMs to allocate to the PM currently being visited, and moves the next PM for allocating the remaining VMs. As a result of the first fit, each of visited child subtrees is assigned with a substring of S_v which is disjoint with other substrings assigned to sibling subtrees. For example, suppose that n VMs are sequentially allocated to the child subtrees$T_{r_1}, \ldots, T_{r_m}$ rooted at r. The subtree T_{r_1} contains a sequence of VMs $\{v_1, v_2, \ldots, v_{d_1}\}$, denoted by $<1, d_1>$. The subtree T_{r_2} has VMs $\{v_{d_1+1}, v_2, \ldots, v_{d_2}\}$, denoted by $<d_1 + 1, d_2>$, and so forth for the subsequent subtrees. Generally, the VMs allocated into the subtree T_{r_i} are represented by the subsequence $<d_{i-1} + 1, d_i>$ $(1 \leq d_{i-1} < d_i < n,\ 1 < i < m)$ and the last used subtree T_{r_m} has $<d_{m-1} + 1, n>$. Thus, essentially FF searches a proper partition of VM sequence S_v in a greedy way such that the resulting substrings can be sequentially allocated to different subtrees. The FF heuristic ensures that if an allocation solution is returned, it must be valid. However, it may neither find a solution nor find optimal one due to its greedy strategy.

In order to find a solution with a better chance and improve its optimality, our algorithm sequentially places VMs with dynamic programming (DP) strategy instead of greedy strategy.

Given a sequence of N VMs, $S_N = \{1, 2, \ldots, N\}$, $<a, b>$ ($a \leq b$) is used to represent the set of VMs in the substring between VM a and VM b, inclusive. Let A_N be the set of all possible substrings of S_N and its size is $1 + \frac{(N+1)N}{2}$ (1 for the empty set). As opposed to the previous DP based allocation algorithm, in this algorithm the allocable VM set M_v for any subtree T_v only consists of all the substrings of S_N that can be allocated into T_v, which means $M_v \subseteq A_N$ and the size of M_v is $O(N^2)$. To compute the allocable VM set for a subtree, each substring in A_N is checked, and the one that can be allocated to the subtree is put into the corresponding allocable VM set. The check procedure is conducted as follows:

(1) For each PM v at level 0 of tree topology T, any substring $<a, b> \in A_N$ can be allocated into v if v has no less than $b - a + 1$ empty slots and the uplink L_v has bandwidth occupancy ratio $O_{L_v}(N, <a, b>) < 1$. $O_{L_v}(N, <a, b>)$ is computed as stated in 5.1, given that L_v divides N into two VM sets $<a, b>$ and $S_N \setminus <a, b>$;

(2) For each vertex v at level $l > 0$, its allocable VM set is calculated in the similar way as line 10-33 of Algorithm 1. But the difference is that, $S_v[i]$ here stores the substrings that could be accommodated in $T_v[i]$. For arbitrary k, $a \leq k \leq b$, if $S_v[i-1]$ and v's i-th child v_i's allocable VM set M_{v_i} include two substrings $<a, k-1>$ and $<k, b>$ respectively (or, $<k, b>$ and $<a, k-1>$ respectively), the substring $<a, b>$ can be accommodated in $T_v[i]$. We define $<a, a-1> = \emptyset$. Then, the substrings in $S_v[m]$ that do not violate the uplink bandwidth constraint are put into v's allocable VM set M_v. Once S_N can be found in M_v, Alloc() is called to allocate S_N into the subtree T_v. The optimization code for bandwidth occupancy ratio is similar to Algorithm 1. $Opt[T_v[i], <a, b>]$ stores the optimal value when $<a, b>$ is assigned to $T_v[i]$. It is obtained by the min operation on maximum bandwidth occupancy ratios over all possible k that have $<a, k-1>$ and $<k, b>$ exist in $S_v[i-1]$ and M_{v_i} separately.

The pseudo code of our algorithm is given in Algorithm 2. It has time complexity $O(|V|\Delta N^4)$. To address stochastic bandwidth demands, the algorithm sorts N VMs by their 95th percentile of their bandwidth demands. Based on the first fit algorithm, It reduces the time complexity of searching feasible allocations by considering the allocatability of only N^2 substrings and optimize the solution with a dynamic-programming strategy. When all the numbers in the VM sequence $<1, 2, \ldots, N>$ are replaced with the same number, which means all VMs are homogeneous, Algorithm 2 is equal to the allocation algorithm for homogeneous SVC in Algorithm 1.

Algorithm 2: Allocation algorithm for heterogeneous model

Input: Datacenter tree topology T, bandwidth reservation information of each link, SVC request $r = \langle N, \mu_1, \sigma_1, \ldots, \mu_N, \sigma_N \rangle$ where $\{(\mu_i, \sigma_i) | 1 \leq i \leq N\}$ are sorted by their 95th percentile in non-ascending order.

1 $A_N \leftarrow$ the set of all the substrings of the sequence $\langle 1, 2, \ldots, N \rangle$;
for *level* $l \leftarrow 0$ *to Height(T)* **do**
2 **for** *each subtree* T_v *rooted at vertex* v *at level* l **do**
3 $m \leftarrow$ the number of v's children;
4 $S_v[0] \leftarrow \{\emptyset\}$;
5 **if** $l = 0$ **then** `// leaf v is a PM`
6 **for** *each substring* $\langle a, b \rangle \in A_N$ **do**
7 **if** *the size of* $\langle a, b \rangle$ *is not larger than* C_v **then**
`// C_v is the number of v's empty slots`
8 $S_v[0] \leftarrow S_v[0] \cup \langle a, b \rangle$;
9 **for** *each substring* $h \in S_v[0]$ **do**
10 $Opt[T_v, h] = 0$; `// No link usage between VMs in the same machine`
11 $T_v[0] \leftarrow v$;
12 **for** *i from 1 to m* **do**
13 $S_v[i] \leftarrow \{\emptyset\}$;
14 $T_v[i] \leftarrow T_v[i-1] \bigoplus T_{v_i}$;
15 **for** *each substring* $\in M_{v_i}$ **do**
16 **for** *each substring* $\in S_v[i-1]$ **do**
17 **if** *pair* $\langle a, k-1 \rangle$ *and* $\langle k, b \rangle$ *exist between* $S_v[i-1]$ *and v's i-th child* v_i*'s allocable set* M_{v_i} **then**
18 Let s be the string $\in S_v[i-1]$ that contributes to $\langle a, b \rangle$;
19 **if** $\langle a, b \rangle$ *is not in* $S_v[i]$ **then**
20 $min_v[i, \langle a, b \rangle] \leftarrow \infty$;
21 $S_v[i] \leftarrow S_v[i] \cup \{\langle a, b \rangle\}$;
22 **if** $i = 1$ **then**
23 $Opt[T_v[i], \langle a, b \rangle] \leftarrow \max\left\{Opt[T_{v_i}, \langle a, b \rangle], O_{L_{v_i}}(N, \langle a, b \rangle)\right\}$
24 **else**
25 $Opt[T_v[i], \langle a, b \rangle] \leftarrow \max\left\{Opt[T_v[i-1], s], Opt[T_{v_i}, \langle a, b \rangle \setminus s], O_{L_{v_i}}(N, \langle a, b \rangle \setminus s)\right\}$
26 **if** $Opt[T_v[i], \langle a, b \rangle] < min_v[i, \langle a, b \rangle]$ **then**
27 $D_v[i, \langle a, b \rangle] \leftarrow (k, s)$;
28 $min_v[i, \langle a, b \rangle] \leftarrow Opt[T_v[i], \langle a, b \rangle]$;
29 **else**
30 $Opt[T_v[i], \langle a, b \rangle] \leftarrow \max\left\{Opt[T_v[i-1], \langle k, b \rangle], Opt[T_{v_i}, \langle a, k-1 \rangle], O_{L_{v_i}}(N, \langle a, k-1 \rangle)\right\}$
31 **if** $Opt[T_v[i], \langle a, b \rangle] < min_v[i, \langle a, b \rangle]$ **then**
32 $D_v[i, \langle a, b \rangle] \leftarrow (k, \langle k, b \rangle)$;
33 $min_v[i, \langle a, b \rangle] \leftarrow Opt[T_v[i], \langle a, b \rangle]$;
34 $M_v \leftarrow \emptyset$;
35 **for** *each substring* $\langle a, b \rangle \in S_v[m]$ **do**
36 **if** $O_{L_v}(N, \langle a, b \rangle) < 1$ **then**
37 $M_v = M_v \cup \{\langle a, b \rangle\}$;
38 $Opt[T_v, \langle a, b \rangle] \leftarrow min_v[m, \langle a, b \rangle]$;
39 **if** $\langle 1, N \rangle \in M_v$ **then**
40 `Alloc` (r, v, N);
41 **return** true;
42 **return** false;

6 WORK-CONSERVING BANDWIDTH GUARANTEE WITH SVC

The purpose of SVC is to take into account the traffic uncertainty during the admission and allocation phase of virtual clusters. The VM allocation based on SVC aims to ensure the bandwidth demands of virtual clusters can be satisfied in a high probability with proper VM placement. It needs to be incorporated with a bandwidth enforcement approach that adaptively adjust the rates according to the bandwidth demands of virtual clusters and the available bandwidth of links. The bandwidth allocation is enforced through rate limiting components of VMs and switches during the run-time phase of virtual clusters. Several approaches [14], [15], [16] have been proposed to adjust the rate to exploit the spare bandwidth from under-utilized reservations to achieve work-conserving bandwidth guarantee. SVC can further improve their efficiency with considering the dynamics of bandwidth demand during the VM allocation phase. In this section, we describe how to incorporate SVC with these approaches and demonstrate the benefit of SVC-based allocation.

In previous sections, we consider the SVC model with the bandwidth demand of normal distribution, which does not provide the minimum bandwidth guarantee. However, we can simply extend the model and allocation algorithms to provide the minimum bandwidth. Still, we use normal distribution to characterize the bandwidth demand but in addition specify the minimum bandwidth guarantee. Then, the homogeneous SVC model can be represented by $\langle N, \mu, \sigma, b\rangle$ where b is the minimum bandwidth guarantee. In the heterogeneous SVC model, each VM has different minimum bandwidth guarantee. Here we focus on the homogeneous SVC model.

In Section 4.1, the condition (2) for valid allocation in previous allocation algorithms needs to be adapted to take into account the minimum bandwidth guarantee. Given a virtual cluster $r_i = \langle N_i, \mu_i, \sigma_i, b_i\rangle$ that has m VMs and $N_i - m$ VMs respectively located in two network components connected by a link L, the minimum bandwidth to be reserved on L is $\min(m, N_i - m)b_i$, denoted by b_i^L. For $K+1$ SVC requests served by link L, their reserved bandwidth on L are $b_1^L, b_2^L, \ldots, b_{K+1}^L$ respectively, and totally $\sum_{i=1}^{K+1} b_i^L$ of bandwidth are reserved. Considering their total bandwidth demand $B^L = \sum_{i=1}^{K+1} B_i^L$ are assumed to follow the normal distribution $\mathcal{N}(\sum_{i=1}^{K+1} \mu_{i,L}, \sum_{i=1}^{K+1} \sigma_{i,L}^2)$, we use truncated normal distribution to characterize the bandwidth demand with minimum guarantee. The truncated normal distribution is a part of the distribution of a normally distributed random variable with value bounded below or above. Here, we consider the one-sided truncated normal distribution of B^L with $B^L > \sum_{i=1}^{K+1} b_i^L$. Its probability distribution should be normalized in order to make sure the total probability over the restricted interval is unity. Let $\mu_L = \sum_{i=1}^{K+1} \mu_{i,L}$ and $\sigma_L = \sqrt{\sum_{i=1}^{K+1} \sigma_{i,L}^2}$. We can compute

$$\Pr(B^L > \sum_{i=1}^{K+1} b_i^L) = 1 - \Phi(\frac{\sum_{i=1}^{K+1} b_i^L - \mu_L}{\sigma_L}) = 1 - \Phi(\alpha^L) \quad (13)$$

where $\alpha^L = \frac{\sum_{i=1}^{K+1} b_i^L - \mu_L}{\sigma_L}$ and $\Phi(\cdot)$ is the standard normal cdf. Then, the truncated normal distribution pdf is

$$f(B^L | B^L > \sum_{i=1}^{K+1} b_i^L) = \frac{f(B^L)}{1 - \Phi(\alpha^L)} = \frac{\frac{1}{\sigma_L}\phi(\frac{B^L - \mu_L}{\sigma_L})}{1 - \Phi(\alpha^L)} \quad (14)$$

Based on this pdf, we can compute the probability $\Pr(B^L > S^L)$ by

$$\Pr(B^L > S^L) = \begin{cases} 1, & \text{if } S^L \le \sum_{i=1}^{K+1} b_i^L \\ \frac{1}{\sigma_L(1-\Phi(\alpha^L))}(1 - \Phi(\frac{S^L - \mu_L}{\sigma_L})), & \text{if } S^L > \sum_{i=1}^{K+1} b_i^L \end{cases} \quad (15)$$

and compare it with ϵ to decide whether it is a valid allocation.

If $S^L \le \sum_{i=1}^{K+1} b_i^L$, it is not a valid allocation. If $S^L > \sum_{i=1}^{K+1} b_i^L$, for a valid allocation, it should have $\frac{1}{\sigma_L(1-\Phi(\alpha^L))}(1 - \Phi(\frac{S^L - \mu_L}{\sigma_L})) < \epsilon$. Then, we can derive that

$$S_L > \mu_L + \sigma_L \Phi^{-1}(1 - \epsilon\sigma_L(1 - \Phi(\alpha^L))) \quad (16)$$

Based on that, we can redefine effective amount of bandwidth in (5) and redefine the bandwidth occupancy ratio in (6) by replacing $\sum_{i=1}^{K} E_i^L$ with the right side of (16). Accordingly, in the allocation algorithms, a set of VMs are allocable only if $S^L > \sum_{i=1}^{K+1} b_i^L$ and the bandwidth occupancy ratio is less than one.

With the extension for minimum bandwidth guarantee, SVC based allocation can be incorporated with existing bandwidth enforcement approaches such as Elasticswitch [16] that provide minimum bandwidth guarantee while exploiting underutilized bandwidth reservation to achieve work-conserving bandwidth allocation. Current work-conserving bandwidth guarantee approaches [14], [15], [16] assume the deterministic horse model and allocate VMs based on that. Instead, the SVC model allows them to take into account the traffic uncertainty during the allocation phase, which can further improve the performance gain achieved by dynamic rate adjustment. Considering an example in Figure 5. The tree topology consists of four PMs each having two VM slots and link capacity 250. There are three virtual clusters $c1$, $c2$ and $c3$ each containing two VMs and requiring the minimum bandwidth of 100. $c3$ is to be allocated, either to the left subtree or to the right subtree. Based on deterministic horse model with minimum bandwidth guarantee 100, the approaches like Elasticswitch can allocate $c3$ to the left subtree, sharing bandwidth resources with $c1$. With SVC model and $\epsilon = 0.05$, $c3$ has to be allocated to the right subtree. This is because that the probability of total bandwidth demand of $c_1$and c_3, denoted by$B_{c1} + B_{c3}$, being larger than 250, is greater than ϵ; instead, the probability of total bandwidth demand $B_{c2} + B_{c3}$ being larger than 250, is less than ϵ. Since the work-conserving approaches increase the rates of VMs to exploit underutilized bandwidth, placing $c3$ in the left subtree creates more competition than in the right subtree, leading to the suboptimal performance.

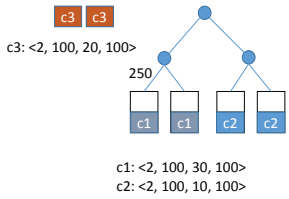


Fig. 5: Example for VM allocation.

7 EVALUATION

We use simulations of large scale datacenter networks to demonstrate the benefits of SVC especially for cloud application workloads with highly volatile bandwidth demands. Our simulation is implemented in Matlab.

7.1 Simulation Setup

We simulate a datacenter of three-level tree topology with no path diversity. A rack consists of 20 machines each with 4 VM slots and a 1Gbps link to connect to a Top-of-Rack (ToR) switch. Every 10 ToR switches are connected to a level-2 aggregation switch and 5 aggregation switches are connected to the datacenter core switch. There are total 1,000 machines at level 0. The default oversubscription of the physical network is 2, which means that the link bandwidth between a ToR switch and an aggregation switch is 10Gbps and the link bandwidth between an aggregation switch and the core switch is 50Gbps.

Workload. The workload models we use for tenant jobs/applications are similar to those in Oktopus [6]. Each job is modeled as a set of tasks to be run on individual VMs and a set of flows of uniform length (L) between tasks. Each task is a source and a destination for one flow. The completion time of a job is $\max(T_c, T_n)$ where T_c is the job's compute time and T_n is the time for the last flow to finish. The number of VMs needed by each job request, i.e., job size, is exponentially distributed around a mean of 49 as in [6]. To simulate the random and dynamic bandwidth demand, we change the data generation rates of the source tasks in a job j every second, which are taken from a normal

distribution $\mathcal{N}(\mu_{d_j}, \sigma^2_{d_j})$. Our SVC is derived from the distribution of data generation rate.

Alternate abstractions. We compare our homogeneous SVC abstraction with the virtual cluster (VC) abstraction $\langle N, B\rangle$ of Oktopus [6]. Given the normal distribution of the data generation rate of a job, we use the mean and 95th percentile of the rate as the requested bandwidth B under VC. The resulting VC models are called *mean-VC* and *percentile-VC*, respectively. We consider them because we believe that they are common ways to accommodate the demand uncertainty under deterministic abstractions. We do not consider TIVC (temporally-interleaved virtual cluster) since our stochastic model does not assume any time-varying patterns.

Bandwidth Enforcement. We consider two different bandwidth enforcement methods that are used with the SVC-based allocation and evaluate the benefit of SVC in these two scenarios. One is a Elasticswitch-like [16] bandwidth enforcement approach which dynamically sets the rate beyond the minimum guarantee according to a weighted TCP-like rate adaption algorithm. In this case, the SVC model with the minimum bandwidth guarantee is used and incorporated with the Elasticswitch-like approach to achieve work-conserving bandwidth guarantee. Based on such bandwidth enforcement, we examine the benefits of SVC for practice. Another is to adjust the rate of VMs proportionally to their bandwidth demands through weighted TCP-like rate adaption algorithm with VMs' bandwidth demands as the weights. It does not enforce the minimum bandwidth guarantee for each virtual cluster, in which case the bandwidth is actually guaranteed in a probabilistic way by proper VM placement. It assumes the perfect prediction to the bandwidth demand of VMs and thus is not practical. The reason we introduce this impractical approach is that, if we compare two alternate deterministic models with the SVC having Elasticswitch-like [16] bandwidth enforcement, we cannot tell whether the performance gain is from the SVC-based allocation or from the Elasticswitch-like bandwidth enforcement. It has demonstrated that Elasticswitch [16] can improve the performance compared with the deterministic models with static bandwidth allocation. Therefore, to effectively investigate the advantages of SVC, we assume this ideal bandwidth enforcement method such that the performance is largely decided by the SVC-based VM allocation.

7.2 Simulation Results

We compare our SVC with Oktopus within two scenarios: first, we consider a large batch of tenant jobs placed in a FIFO queue waiting to be allocated to run; second, tenant jobs dynamically arrive over time and are accepted only if they can be allocated at the moment of arrival. In each scenario, we simulate 500 tenant jobs. The compute time of each job is randomly chosen from $[200, 500]$. For the data generation rate of job j, μ_{d_j} is randomly chosen from $\{100, 200, 300, 400, 500\}$ and σ_{d_j} is $\rho\mu_{d_j}$ where ρ is a deviation coefficient to reflect the degree of the traffic demand uncertainty and its value is randomly chosen form (0,1) by default. The default risk factor ϵ is 0.05. To investigate the potential benefits of SVC, we consider the SVC with no minimum bandwidth guarantee and compare it with two alternate models. Therefore, we choose the second bandwidth enforcement method in our simulation.

7.2.1 Batched jobs

For batched jobs, we use the same job scheduling policy as in [6]: jobs are placed in a FIFO queue, and once a job completes, the topmost job(s) that can be allocated is scheduled to run. Fig. 6 shows the total completion time of 500 jobs in a batch. Fig. 7 shows the average running time per job with increasing the deviation coefficient ρ in order to increase the demand variance. Comparing these two figures, we note that mean-VC has the best performance over other models for the total completion time of batched jobs, but have the worst performance for the average running time per job, which is critical for on-line processing for dynamically arriving jobs. This is because that when large increase of data traffic appears, the fixed bandwidth demand reserved by mean-VC for each VM becomes the bottleneck, which further increases the flow latency and the job completion time. Other models reserve extra bandwidth, and thus incur less running time per job. Compared with mean-VC, percentile-VC is just the opposite. Percentile-VC reserves 95th percentile amount of bandwidth with considering the demand distribution, thus it achieves constant and smallest running time under different deviation coefficients in Fig.7. However, the 95th percentile bandwidth has to be reserved, which can be large especially under high demand variance. Because percentile-VC reserves large amount of bandwidth for each job, the job concurrency in the datacenter is greatly limited, causing that percentile-VC has worst performance for batched jobs in the total completion time. In contrast, mean-VC reserves smallest bandwidth compared with others and has largest concurrent jobs which effectively reduces the total completion time.

The comparison results above between percentile-VC and mean-VC actually shows the trade-off between the job concurrency and job running time. The increase in reserved bandwidth reduces the flow latency and thus the job running time, but also decreases the job concurrency, causing longer waiting time of jobs. However, both the percentile-VC and mean-VC cannot achieve such trade-off. But as we can see from Fig.6 and 7, our SVC model actually achieves the trade-off. Compared with Percentile-VC, SVC significantly increases the job concurrency indicated by smaller total completion time of batched jobs, while obtaining comparable job running time which is much less than mean-VC. The reason for the improvement is that, in SVC, multiple jobs share the bandwidth with total bandwidth demand not exceeding the link capacity with a high probability (0.95 when $\epsilon = 0.05$), which ensures that sufficient bandwidth is reserved statistically while each job can also exploit the unused bandwidth. With smaller ϵ, SVC provides better bandwidth guarantee and thus smaller job running time but reduces the job concurrency, which means that we can tune ϵ to achieve the desired trade-off.

7.2.2 Dynamically arriving jobs

In cloud the tenants requests usually arrive over time. In our simulation the job arrival follows a poisson process with rate λ, then the load on a datacenter with M total VMs is $\frac{\lambda N T_c}{M}$ where N is the mean job size and T_c is the mean compute time. As in previous works [6], [7], if a job cannot be allocated upon its arrival, it is rejected. We compare the job rejection rates under SVC with different risk factor $\epsilon = 0.02, 0.05$, as well as the mean-VC and percentile-VC. In Fig. 8, we can see that under low load of 20%, all methods have zero or close to zero request rejection rates. As the load increases, they have relationship mean-VC < SVC($\epsilon = 0.05$) < SVC($\epsilon = 0.02$)< percentile-VC. With considering the demand uncertainty, the effective bandwidth amount SVC reserves is larger than the mean of bandwidth demand, as we can see from (5), thus SVC has higher rejection rates than mean-VC. For SVC, smaller risk factor ϵ incurs higher bandwidth reservation, and causes higher rejection rate, which also indicates

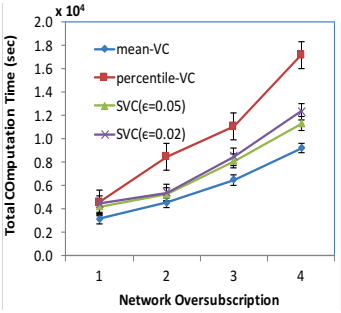


Fig. 6: Completion time with varying oversub.

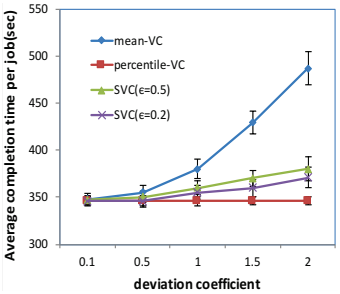


Fig. 7: Average completion time per job with varying deviation coefficient.

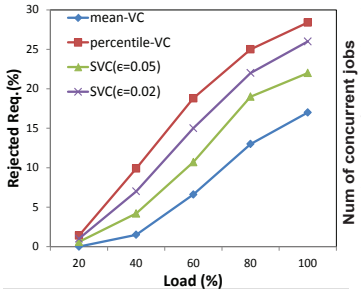


Fig. 8: Percentage of rejected requests with varying datacenter load.

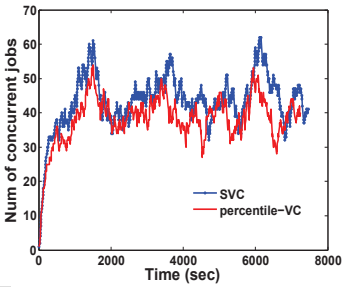


Fig. 9: Concurrent jobs for 60% load.

that $\epsilon = 0.02$ provides stronger bandwidth guarantee to stochastic bandwidth demands. For percentile-VC, combined with the results of job completion times stated earlier, we can see that percentile-VC achieves similar completion time to SVC($\epsilon = 0.05$) but at the cost of higher rejection rate. The reason is because it provides deterministic but exclusive bandwidth to accommodate the varying traffic demands. Instead, SVC uses the bandwidth statistically shared with other tenant jobs under probabilistic bandwidth constraint (1), so it is possible to have higher job concurrency. To understand this, we record the number of concurrent jobs under percentile-VC and SVC ($\epsilon = 0.05$) when a new job arrives at the system. Fig. 9 show that SVC consistently achieves about 10% higher job concurrency than percentile-VC.

7.2.3 *Allocation Algorithms*

In this section we compare our allocation algorithms for homogeneous SVC models with an adapted allocation algorithm proposed for TIVC [7] to demonstrate the benefit of minimizing the maximum bandwidth occupancy ratio described in Section 4.2.2 . In the following, we simply use SVC-alloc and TIVC-alloc to refer the two allocation algorithms. It is worth to note that here we are not comparing two different network abstractions. SVC-alloc and TIVC-alloc are compared in terms of the performance for allocating SVC.

We adapt the TIVC's allocation algorithm [7] to serve SVC allocation, by replacing the condition of valid allocation by ours shown in (2) for computing the allocable VM sets, and additionally maintaining the information of the stochastic bandwidth demands of every SVC allocation on each link (e.g., the means and variances of $B_1^L, \ldots, B_K^L$ in Fig. 2). In this way, the key difference between the TIVC-alloc algorithm and our SVC-alloc is that TIVC-alloc does not consider the optimization on bandwidth occupancy.

We assume the cloud scenario for dynamically arriving jobs as above to evaluate the performance of two algorithms on the link bandwidth occupancy. We sampled the maximum of bandwidth occupancy ratios among all links every time when a new job arrives. Fig. 10 shows the empirical cumulative probability distribution of maximum bandwidth occupancy ratio in the datacenter under different loads 20% and 60% for our homogeneous SVC allocation algorithm and adapted TIVC algorithm. From the figure we can easily see that SVC-alloc achieves better bandwidth occupancy overhead than TIVC-alloc under both loads. Under load 20%, among all the maximum bandwidth occupancy ratios, SVC-alloc has 50% samples less than 0.996 but TIVC-alloc has only about 10%. When the load increases to 60%, the link bandwidth occupancy becomes higher for both algorithms. SVC-alloc and TIVC-alloc have 80% and 95% of maximum bandwidth occupancy ratios distributed on [0.997,1], respectively.

We further evaluate the request rejection rates of SVC-alloc and TIVC-alloc under different loads and show the result in Fig. 11. We note that SVC-alloc and TIVC-alloc have almost the same rejection rates, which means the optimization of link bandwidth occupancy in SVC-alloc affects little on the optimization goal of TIVC-alloc to maximize the ability to accommodate future tenant requests.

Besides, we compared SVC-alloc and TIVC-alloc algorithms in terms of their running time. SVC-alloc has the same time complexity $O(|V|\Delta N^2)$ as TIVC-alloc that has been demonstrated to achieve high scalability in previous works [7]. Following the same methodology to evaluate scalability, we measured the running time of SVC-alloc and TIVC-alloc algorithms for every arriving job on 4-core Intel i5 3.4Ghz processor and 8GB RAM. For SVC-alloc, the median time is 1.8 seconds, 95th percentile is 38 seconds, 99th percentile is 60 seconds in our simulation settings. TIVC-alloc has similar running time with median time 1.7s, 95th percentile 36s and 99th percentile 60s, because both SVC-alloc and TIVC-alloc follow the same dynamic programming procedure. The algorithm implementations with using other languages like C will further shorten the running time. For the applications that do not require instant deployment, like batch processing jobs for big data that runs at the order of magnitude of minutes or hours, the running time of allocation algorithms at the order of magnitude in seconds may be negligible. However, when the job arrival rate is high, the system can be very sensitive to the running time of allocation algorithms, because scheduling one big job may delay scheduling small jobs and cause those small jobs backlogged. In this case a global cluster scheduler may optimize its scheduling strategy accordingly with considering this scheduling delay.

We also compared our heterogeneous SVC allocation algorithm (referred to as SVC-H) with the first-fit (FF) algorithm that does not conduct dynamic programming optimization, with using the similar simulation settings. The distribution of maximum bandwidth occupancy ratio and request rejection rates are shown in Fig. 13 and 14 respectively. As we can see, SVC-H and FF have similar relationship to the relationship between SVC-alloc and TIVC-alloc as shown in Fig. 10 and 11. With the optimization on the bandwidth occupancy, SVC-H algorithm achieves better bandwidth occupancy overhead and lower rejection rates compared with FF algorithm. The running time of SVC-H has the median 5.74s, 95th percentile is 68s, 99th percentile is 108s, 2-3 times higher than SVC-alloc, since SVC-H in Algorithm 2 has time complexity $O(|V|\Delta N^4)$ that has extra $O(N^2)$ factor than SVC-alloc in Algorithm 1.

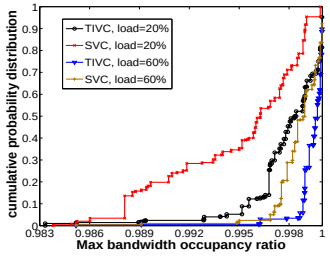


Fig. 10: The Cumulative probability distribution(CDF) of sampled max bandwidth occupancy ratio.

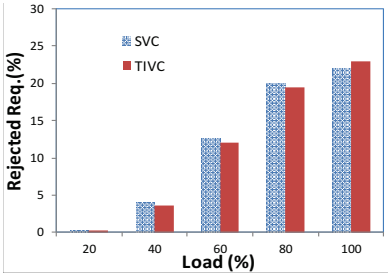


Fig. 11: Percentage of rejected requests with varying datacenter load.

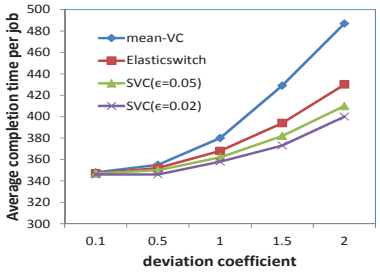


Fig. 12: Average completion time per job under work-conserving bandwidth guarantee.

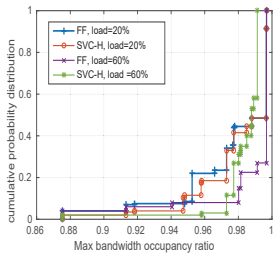


Fig. 13: The CDF of sampled max bandwidth occupancy ratio.

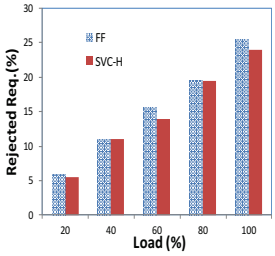


Fig. 14: Percentage of rejected requests with varying datacenter load.

7.2.4 *Improvement to Work-Conserving Bandwidth Guarantee*

Based on the SVC model and allocation algorithms with minimum bandwidth guarantee given in Section 6, we consider a work-conserving bandwidth guarantee approach that combines the SVC-based allocation and Elasticswitch-like bandwidth enforcement. We compare it with Elasticswitch that allocates VMs based on the deterministic hose model, and show the performance improvement of SVC-based allocation in an online scenario with dynamically arriving jobs. We use the same simulation settings in the above section and in addition we set the the mean of the normal distribution as the minimum bandwidth guarantee for a virtual cluster. We assume a light load of 20% such that the rejected request ratio is negligible and sufficient spare bandwidth can be exploited by adaptive rate adjustment for work-conserving bandwidth enforcement. Figure 12 shows the average job completion time with using different bandwidth guarantee approaches. The difference between mean-VC and Elasticswitch with the mean as the minimum bandwidth is that Elasticswitch can adaptively increase the rate beyond the static reservation of mean-VC. They have the same VM allocation algorithms. It is obvious that Elasticswitch can increase the throughput of the tasks and thus reduce the average job completion time compared with mean-VC, as shown in Figure 12. This result is consistent with the previous work [16]. The figure also shows that the SVC-based approach with $\epsilon = 0.05$ and $\epsilon = 0.02$ have lower average job completion time than Elasticswitch. The reason for that has been demonstrated by Figure 5. SVC-based approaches allocate virtual clusters with considering the traffic variance and avoid the bottlenecks during the VM placement. In this way, it reduces the bandwidth competition and increases the efficiency of adaptive rate control for exploiting the spare bandwidth.

8 Conclusion and Future Work

In this paper we explore a new virtual network abstraction, SVC, which models the uncertainty of bandwidth demands of cloud applications. Based on SVC, we introduce a VM allocation framework which provides probabilistic bandwidth guarantee to the tenants. To enforce the framework, we propose dynamic programming based VM allocation algorithms which not only achieve good locality of VMs but also minimize the maximum of bandwidth occupancy ratio on links. Simulation results demonstrate that SVC yields better performance for cloud application workloads with highly volatile bandwidth demands, and achieves the trade-off between the job concurrency and the job running time. Besides, we propose to extend SVC with minimum bandwidth guarantee and incorporate it with existing work-conserving bandwidth enforcement approaches. We show its improvement to the work-conserving bandwidth guarantee. For simplicity, we assume normal distribution for the bandwidth demand in this paper, but SVC can straightforwardly use other types of probability distributions.

Note that SVC can be extended to characterize the time varying bandwidth demand which means that the distribution of bandwidth demand can change with time. As TIVC that specifies different deterministic bandwidth demand during different time intervals, we can extend SVC by allowing specific probability distributions for VM bandwidth demand during different time intervals. For normal distribution, time varying SVC can be represented $\langle N, (T_{i1}, T_{i2}, u_i, \sigma_i) | 1 <= i <= I \rangle$ where each time interval $[T_{i1}, T_{i2}]$ is associated with a normal distribution of VM bandwidth demand $\mathcal{N}(u_i, \sigma_i)$. In this case, link L that carries the traffic of this SVC should have enough residual bandwidth during any time interval $[T_{i1}, T_{i2}]$. The inequality (1) should hold for the VM bandwidth distributions of every virtual cluster across link L during each of their time intervals. In this paper, we focus on simple SVC model and leave time varying SVC in our future work. Our future work also includes characterizing the probability distributions of bandwidth demands from a variety of real workloads, and implementing and evaluating SVC in a real cloud environment.

Acknowledgements

This research was supported in part by U.S. NSF grants NSF 1547102, SaTC 1564097, CRISP 1541074, SAVI/RCN 1550379 and 1402266, CNS 1421561, NSF-1404981, IIS-1354123, CNS-1254006, CNS-1249603, OAC-1724845, ACI-1719397, CNS-1733596, Microsoft Research Faculty Fellowship 8300751, IBM Faculty Award 2015-2017. An early version of this work was presented in the Proceedings of ICDCS 2014 [25].

REFERENCES

[1] "Hadoop," hadoop.apache.org, 2016, [Online; accessed 19-August-2017].

[2] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: observing, analyzing, and reducing variance," *Proc. of VLDB Endow.*, vol. 3, no. 1-2, pp. 460–471, Sep. 2010.

[3] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris, "Reining in the outliers in map-reduce clusters using mantri," in *Proc. of ODSI*. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–16.

[4] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, "Sharing the data center network," in *Proc. of the 8th USENIX conference on Networked systems design and implementation (NSDI)*, Berkeley, CA, USA, 2011, pp. 23–23.

[5] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: a data center network virtualization architecture with bandwidth guarantees," in *Proc. of Co-NEXT*. New York, NY, USA: ACM, 2010, pp. 15:1–15:12.

[6] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proc. of ACM SIGCOMM*. New York, NY, USA: ACM, 2011, pp. 242–253.

[7] D. Xie, N. Ding, Y. C. Hu, and R. R. Kompella, "The only constant is change: incorporating time-varying network reservations in data centers," in *Proc. of ACM SIGCOMM*, 2012, pp. 199–210.

[8] J. Lee, Y. Turner, M. Lee, L. Popa, S. Banerjee, J.-M. Kang, and P. Sharma, "Application-driven bandwidth guarantees in datacenters," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. New York, NY, USA: ACM, 2014, pp. 467–478.

[9] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 92–99, Jan. 2010.

[10] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," in *Proc. of ACM IMC*. New York, NY, USA: ACM, 2009, pp. 202–208.

[11] D. Mitra and Q. Wang, "Stochastic traffic engineering for demand uncertainty and risk-aware network revenue management," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 221–233, Apr. 2005.

[12] J. Cao, D. Davis, S. V. Wiel, B. Yu, S. Vander, and W. B. Yu, "Time-varying network tomography: Router link data," *Journal of the American Statistical Association*, vol. 95, pp. 1063–1075, 2000.

[13] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and new directions," in *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '02. New York, NY, USA: ACM, 2002, pp. 161–174.

[14] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes, "Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks," in *Proceedings of the 3rd Conference on I/O Virtualization*, ser. WIOV'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 6–6.

[15] V. Jeyakumar, M. Alizadeh, D. Mazires, B. Prabhakar, and C.Kim, "Eyeq: Practical network performance isolation for the multi-tenant cloud," in *Proc. of Usenix HotCloud*, 2012.

[16] L. Popa, P. Yalagandula, S. Banerjee, J. C. Mogul, Y. Turner, and J. R. Santos, "Elasticswitch: Practical work-conserving bandwidth guarantees for cloud computing," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 351–362.

[17] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "Faircloud: sharing the network in cloud computing," in *Proc. of ACM SIGCOMM*. New York, NY, USA: ACM, 2012, pp. 187–198.

[18] V. T. Lam, S. Radhakrishnan, R. Pan, A. Vahdat, and G. Varghese, "Netshare and stochastic netshare: Predictable bandwidth allocation for data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, pp. 5–11, Jun.

[19] J. Zhu, D. Li, J. Wu, H. Liu, Y. Zhang, and J. Zhang, "Towards bandwidth guarantee in multi-tenancy cloud computing networks," in *Proc. of IEEE ICNP*, 2012, pp. 1–10.

[20] L. Yu and Z. Cai, "Dynamic scaling of virtual clusters with bandwidth guarantee in cloud datacenters," in *IEEE INFOCOM*, April 2016, pp. 1–9.

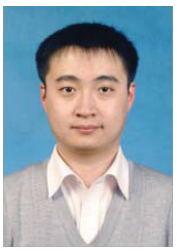
[21] J. Guo, F. Liu, J. Lui, and H. Jin, "Fair network bandwidth allocation in iaas datacenters via a cooperative game approach," *TON*, 2015.

[22] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. of IEEE INFOCOM*, 2010, pp. 1–9.

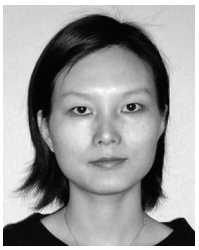
[23] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proc. of IEEE INFOCOM*, 2011, pp. 71–75.

[24] S. Nadarajah and S. Kotz, "Exact distribution of the max/min of two gaussian random variables," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, no. 2, pp. 210–212, 2008.

[25] L. Yu and H. Shen, "Bandwidth guarantee under demand uncertainty in multi-tenant clouds," in *Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems*, ser. ICDCS '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 258–267.



Lei Yu received his BS degree and MS degree in computer science from Harbin Institute of Technology, China. He is a Ph.D student in the School of Computer Science at Georgia Institute of Technology. His research interests include cloud computing, big data, privacy, sensor networks, wireless networks, and network security.



Haiying Shen received the BS degree in Computer Science and Engineering from Tongji University, China in 2000, and the MS and Ph.D. degrees in Computer Engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Associate Professor in the Computer Science Department at the University of Virginia. Her research interests include distributed computer systems and computer networks, cloud computing and cyber-physical systems. She is a Microsoft Faculty Fellow of 2010, a senior member of the IEEE and a member of the ACM.

Zhipeng Cai received his PhD and MS degrees from the Department of Computing Science at University of Alberta, and BS degree from the Department of Computer Science and Engineering at Beijing Institute of Technology. He is currently an Assistant Professor in the Department of Computer Science at Georgia State University. Prior to joining GSU, Dr. Cai was a research faculty in the School of Electrical and Computer Engineering at Georgia Institute of Technology. Dr. Cai's research areas focus on Networking and Big data. Dr. Cai is the recipient of an NSF CAREER Award.

Ling Liu is a Professor in the School of Computer Science at Georgia Institute of Technology. She directs the research programs in Distributed Data Intensive Systems Lab (DiSL), examining various aspects of large-scale data intensive systems, including performance, availability, security, privacy and trust. Prof. Liu is an elected IEEE Fellow, a recipient of IEEE Computer Society Technical Achievement Award (2012). She has published over 300 international journal and conference articles and is a recipient of the best paper award from a number of top venues, including ICDCS, WWW, 2005 Pat Goldberg Memorial Best Paper Award, IEEE Cloud, IEEE ICWS, ACM/IEEE CCGrid, IEEE ICIOT, IEEE EDGE. In addition to serve as general chair and PC chairs of numerous IEEE and ACM conferences in distributed computing, cloud computing and database fields, Prof. Liu has served on editorial board of over a dozen international journals, including the editor in chief of IEEE Transactions on Service Computing (2013-2016). Her current research is primarily sponsored by NSF and IBM.

Calton Pu received his PhD from University of Washington in 1986 and served on the faculty of Columbia University and Oregon Graduate Institute. Currently, he is holding the position of Professor and John P. Imlay, Jr. Chair in Software in the College of Computing, Georgia Institute of Technology. He has worked on several projects in systems and database research. He has published more than 70 journal papers and book chapters, 200 conference and refereed workshop papers. He served on more than 120 program committees. His recent research has focused on big data in Internet of Things, automated N-tier application deployment and denial of information. He is an elected IEEE Fellow and a member of the ACM.