Bandwidth Guarantee under Demand Uncertainty in Multi-tenant Clouds

Lei Yu and Haiying Shen
Department of ECE, Clemson University, Clemson, SC, USA
{leiy, shenh}@clemson.edu

Abstract—**The shared multi-tenant nature of cloud network infrastructures has caused poor application performance in the clouds due to unpredictable network performance. To provide bandwidth guarantee, several virtual network abstractions have been proposed which allow the tenants to specify and reserve virtual clusters with required network bandwidth between the VMs. However, all of these existing proposals require the tenants to deterministically characterize the exact bandwidth demands in the abstractions, which can be difficult and result in inefficient bandwidth reservation due to the demand uncertainty. In this paper, we propose a virtual cluster abstraction with stochastic bandwidth requirements between VMs, called Stochastic Virtual Cluster (SVC), which probabilistically models the bandwidth demand uncertainty. Based on SVC, we propose a network sharing framework and efficient VM allocation algorithms to ensure that the bandwidth demands of tenants on any link are satisfied with a high probability, while minimizing the bandwidth occupancy cost on links. Using simulations, we demonstrate the effectiveness of SVC for accommodating cloud application workloads with highly volatile bandwidth demands, in the way of achieving the trade-off between the job concurrency and average job running time.**

I. Introduction

Cloud computing, based on modern virtualization of datacenters, delivers cost-effective and powerful Infrastructure as a Service (IaaS) for a wide spectrum of applications like those which use the MapReduce paradigm to process large data sets [1]. However, the shared, multi-tenant nature of cloud network infrastructures have caused unpredictable application performance in the cloud [2] due to the competition of applications for the scarce network resources. The lack of guaranteed network bandwidth leads to variable data transmission latency and job completion time, causing poor job scheduling and datacenter throughput [3], [4].

To provide the network performance guarantees, several recent works such as SecondNet [5], Oktopus [6], and TIVC [7] have proposed virtual network abstractions which explicitly enable bandwidth reservation in datacenter networks. SecondNet aims to guarantee end-to-end bandwidth for each pair of virtual machines (in short, VMs). Oktopus proposes a virtual cluster model that requires a virtual topology comprising N VMs connected by links to a virtual switch with a specified constant bandwidth capacity. With the observation that many cloud applications have time-varying bandwidth requirements, TIVC further proposes a temporally-interleaved virtual cluster model that allows specifying the different bandwidth demands at different times. All these methods allocate VM slots on the physical machines with the goals of satisfying the bandwidth requirement of virtual network abstractions, as well as achieving good locality of VMs. Good locality means that the VMs allocated for a job should be as localized to a subtree/cluster as possible, to conserves the bandwidth of the links and maximize the ability to accommodate future tenant requests. While these methods enable the bandwidth reservation for tenants, they require reliable and deterministic estimate of bandwidth demand among VMs; such estimate is made with the workload pattern obtained either from the understanding of the cloud applications or from profiling runs [7].

Unfortunately, recent measurement studies [8], [9] show that the network traffic is highly volatile in production datacenters. It is difficult for a tenant to determine the exact amount of bandwidth it needs at a particular time under demand uncertainty. Although the usual approach of over-provisioning can resolve the problem, it leads to significant resource waste to cloud providers and high costs to tenants. Thus, it is imperative to find efficient bandwidth reservation methods with consideration of uncertain bandwidth demands.

The primary contribution of this paper is to explore a new network reservation abstraction called *Stochastic Virtual Cluster* (SVC) that allows probabilistic characterization of bandwidth demands in virtual networks. The model can better capture the uncertainty and variance of the bandwidth demands of cloud applications by the probabilistic description of bandwidth requirement. Based on the SVC model, we propose a stochastic network sharing framework that provides probabilistic bandwidth guarantee to tenants, i.e., reserves sufficient bandwidth on any links such that the bandwidth demands of all tenants are satisfied with a high probability. With accommodating the bandwidth demand uncertainty, the stochastic network sharing framework provides better performance guarantee to cloud applications with highly volatile bandwidth demands. Also, SVC allows the sharing of the bandwidth among multiple stochastic demands, which improves the job concurrency in clouds.

To enforce our network sharing framework, a fundamental problem is how to allocate VMs in a physical datacenter for a SVC abstraction while ensuring the probabilistic bandwidth guarantee for all tenants. To solve the problem, as opposed to previous works for the allocation of deterministic virtual clusters [6], [7], we need to characterize the bandwidth reservation on any links 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, is not optimal in terms of bandwidth occupancy cost. Thus, in this paper, we first derive the condition for a valid VM allocation that provides probabilistic bandwidth guarantee, and then propose a dynamic programming based algorithm which finds the optimal VM allocation within the lowest-level subtree while minimizing the maximum bandwidth occupancy ratio of the links in the subtree. 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 network sharing framework which provides probabilistic bandwidth guarantee to tenants.
- We devise efficient VM allocation algorithms for SVC with homogenous and heterogeneous bandwidth demands, which not only achieve good locality but also minimize the bandwidth occupancy cost of links.
- We demonstrate the effectiveness of SVC and our network sharing solution 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 II describes the related work. Section III describes our SVC model and the stochastic network sharing framework. Section IV and Section V present the VM allocation algorithms for SVC with homogenous and heterogeneous bandwidth demand, respectively. Section VI shows our simulation results.

II. Related Work

Several network sharing solutions for multi-tenant datacenters have been proposed [4]–[7], [10], [11]. These solutions have different sharing policies, including weight proportional bandwidth sharing [4], [10], [11] and bandwidth reservation [5]–[7].

Seawall [4], Netshare [11] and FairCloud [10] share the network bandwidth among VMs, services or tenants based on weights. Seawall [4] proposes a VM-level congestion control approach to ensure the share of bandwidth obtained by per source VM along all network links is proportional to its weight. Netshare [11] proposes a hierarchical weighted max-min fair sharing mechanism which allocates the bandwidth to services according to their weights and then allocates the bandwidth of each service equally to its TCP connections. Faircloud [14] describes a comprehensive set of properties for cloud network sharing and proposes the allocation policies to navigate the tradeoff space among these properties. These solutions provide proportional fair sharing of networks with minimum bandwidth guarantee and statistical multiplexing,

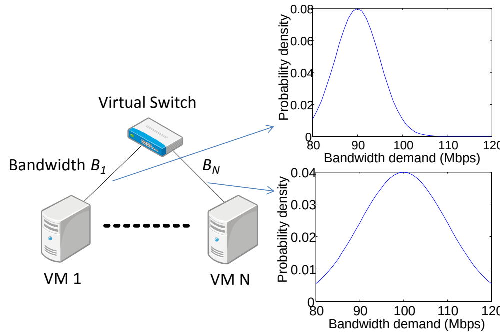


Fig. 1: Virtual cluster model with stochastic bandwidth demand.

but cannot provide deterministic bandwidth guarantees. The performance still depends on other tenants.

Secondnet [5] and Oktopus [6] provide bandwidth guarantee through deterministic bandwidth reservations in the network. Secondnet [5] proposes a virtual datacenter abstraction with the bandwidth requirements among VM pairs. Oktopus [6] proposes a virtual cluster abstraction for network reservation, in which a virtual cluster request $< N, B >$ requires a virtual topology comprising N machines connected by links of bandwidth B to a virtual switch. TIVC [7] extends the virtual cluster model with time-varying bandwidth reservation in order to capture the time-varying networking requirement of cloud applications. The work in [12] extends the virtual cluster model with allowing heterogeneous bandwidth requirements for different VMs. In these works various VM allocation algorithms are proposed for bandwidth guaranteed virtual to physical mapping. Though reservation based mechanisms may not make full utilization of the network bandwidth, it provides predictable network performance for a tenant. However, all these works require deterministic bandwidth demand information which may be difficult to estimate due to demand uncertainty. Our work addresses this problem by extending the virtual cluster model with stochastic bandwidth demand.

III. STOCHASTIC CLOUD NETWORK SHARING

In this section we propose a new virtual cluster network abstraction with stochastic bandwidth demand to account for demand uncertainty, and propose a stochastic cloud network sharing framework that provides bandwidth guarantee in a probabilistic way.

A. Stochastic Virtual Cluster Model

Highly dynamic bandwidth usage of cloud applications [8], [9] 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 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 we will show in our experiments. 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.

Given the bandwidth usage profile of an application, one can derive the probability distributions of bandwidth demands of VMs and include them in the virtual cluster requests. In this paper, to characterize the bandwidth demand uncertainty, we assume that the bandwidth demand B follows a normal distribution $\mathcal{N}(\mu, \sigma^2)$ as in [13], [14], with mean $\mu = E(B)$ and variance $\sigma = Var(B)$. Then, a virtual cluster request has a format of $< N, (\mu_1, \sigma_1), (\mu_2, \sigma_2), \ldots, (\mu_N, \sigma_N) >$, in which N is the number of VMs and the bandwidth demand for each VM i follows normal distribution $\mathcal{N}(\mu_i, \sigma_i^2)$.

The SVC model is reduced to traditional *deterministic virtual cluster* model [6] if $\mu_1 = \mu_2 \ldots = \mu_N$ and $\sigma_i = 0, \forall i$. Indeed, the tenants can either specify the constant bandwidth requirement as in the traditional virtual cluster proposed in [6], or make the stochastic bandwidth demand by giving the probability distributions in SVC. The deterministic and stochastic bandwidth requirements can co-exist in the datacenters in our network sharing framework.

B. Probabilistic Bandwidth Guarantee for Stochastic Bandwidth Demand

For a virtual cluster with deterministic link bandwidth, the bandwidth to be reserved on physical links is fixed given the placement of VMs. The bandwidth is guaranteed through ensuring sufficient bandwidth on the physical links that connect VMs and enforcing the bandwidth reservation by rate limiting on VMs or switches. However, the problem of bandwidth guarantee for the SVC model is different due to the stochastic bandwidth requirement, and accordingly the notion called *probabilistic bandwidth guarantee* is introduced.

We first characterize the bandwidth allocation of K already allocated requests with stochastic bandwidth demands on an underlying physical link L. Let C_L be the bandwidth capacity of link L. Since in our cloud network sharing framework the deterministic and stochastic bandwidth requirements can coexist, a deterministic portion of link bandwidth is reserved for the deterministic virtual cluster requests, the residual bandwidth is sharing among the stochastic virtual clusters. Denote the total amount of the reserved deterministic bandwidth demand on link L by D_L. Then, as shown in Fig. 2, the residual bandwidth $S_L = C_L - D_L$ is sharing among K stochastic virtual clusters, called *stochastic sharing bandwidth*.

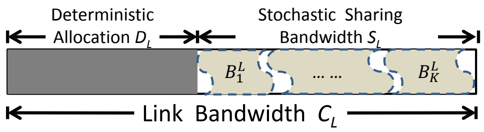

Deterministic Allocation D_L
Stochastic Sharing Bandwidth S_L
B_1^L
… …
B_K^L
Link Bandwidth C_L

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

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. Here probabilistic bandwidth guarantee is provided to these K virtual clusters, in the sense that link L can satisfy their bandwidth demands with a high probability $1 - \epsilon$, i.e.,

$$\Pr(\sum_i B_i^L > S_L) < \epsilon. \tag{1}$$

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 the bandwidth shortage with regard to the tenants' demands. It can be determined by the cloud provider as a part of a service level agreement.

C. Cloud Network Sharing with Probabilistic Bandwidth Guarantee

With the stochastic virtual cluster abstraction, we propose a stochastic cloud network sharing framework for multi-tenant datacenters that provides bandwidth guarantee in a probabilistic way.

Like previous works [6], [7], our network sharing framework relies on two components, network manager and rate limiting components, to implement the SVC models. A network manager, upon receiving a tenant request, performs admission control and VM allocation in the datacenter with physical links satisfying the bandwidth requirements in term of the probabilistic constraint (1). Rate limiting components at endhost hypervisors or switches are used to enforce the bandwidth reservations by ensuring that VMs do not exceed the bandwidth specified in the virtual topology. But as opposed to previous works, our framework uses the rate limiting component to enforce the bandwidth reservation for requests with deterministic bandwidth demands. Since SVC statistically shares the bandwidth on any links among virtual clusters with stochastic bandwidth demands, the variance of bandwidth demand is allowed and no fixed bandwidth reservation needs to be enforced for them. Instead, proper VM placement is required to ensure probabilistic bandwidth guarantee for stochastic bandwidth demands.

Formally, the *VM allocation problem* in our network sharing framework is to allocate N empty VM slots to the tenant's request $< N, (\mu_1, \sigma_1), (\mu_2, \sigma_2), \ldots, (\mu_N, \sigma_N) >$ such that each physical link can still satisfy the constraint (1) for all stochastic bandwidth demands it carries.

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 machines; (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. For designing the allocation algorithms, we focus on tree-like topologies such as multi-rooted tree topologies used in today's datacenters. 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 the following sections we present our VM allocation algorithms.

IV. 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 the bandwidth demand of every VM follows the same probability distribution, i.e., for a SVC request $< N, (\mu_1, \sigma_1), (\mu_2, \sigma_2), \ldots, (\mu_N, \sigma_N) >$, $\mu_i = \mu$, $\sigma_i = \sigma$, $\forall i$. In brief, we use the format $< N, \mu, \sigma >$ for such request. The VM allocation problem has been shown to be NP-hard for deterministic virtual cluster model [6], [12]. To solve the problem, we first determine the condition of valid VM allocation for stochastic bandwidth demands, and based on that we propose approximate allocation algorithms with the goal to achieve both good locality of VMs and less bandwidth occupancy on links.

A. Characterizing bandwidth demand

We begin with characterizing the stochastic bandwidth demand of an allocation for a SVC request $r =< N, \mu, \sigma >$ on a link. For simplicity, we assume that the bandwidth demands of N VMs, denoted by $B_1, \ldots, B_N$, are independent and identically distributed (i.i.d.) random variables. 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. Since each VM has an independent random bandwidth demand following the normal distribution $\mathcal{N}(\mu, \sigma^2)$, the aggregate bandwidth demand of m VMs, denoted by $\mathcal{B}(m)$, has distribution $\mathcal{N}(m\mu, m\sigma^2)$. Accordingly, the total bandwidth demand on link L for request r, denoted by $B_r^L(m)$, is $\min(\mathcal{B}(m), \mathcal{B}(N-m))$. It is the min of two normal variables, of which the exact distribution can be found in [15]. Based on [15], we can easily derive the following lemma:

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

$$E(X) = \mu_1\Phi(\alpha) + \mu_2\Phi(-\alpha) - \theta\phi(\alpha) \tag{2}$$

$$Var(X) = (\sigma_1^2+\mu_1^2)\Phi(\alpha)+(\sigma_2^2+\mu_2^2)\Phi(-\alpha)-(\mu_1+\mu_2)\theta\phi(\alpha)-(E(X))^2 \tag{3}$$

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 $\alpha = \frac{\mu_2 - \mu_1}{\theta}$.

According to Lemma 1, we can compute the mean and variance of $B_r^L(m) = \min(\mathcal{B}(m), \mathcal{B}(N-m))$, denoted by $\mu_{r,L}$ and $\sigma_{r,L}^2$, respectively.

B. Determining valid allocation

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 request r_i has bandwidth demand B_i^L on link L with the mean $u_{i,L}$ and the variance $\sigma_{i,L}^2$. 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 under the allocations for different requests, 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. 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

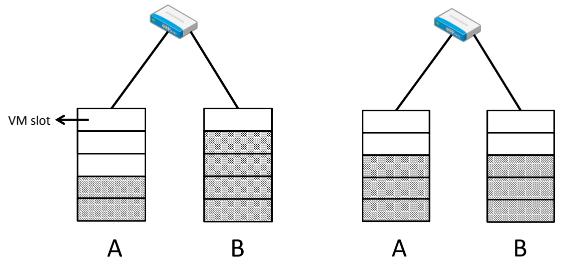
$$\frac{S_L - \sum\limits_{i=1}^{K+1} \mu_{i,L}}{\sqrt{\sum_{i=1}^{K+1} \sigma_{i,L}^2}} > \Phi^{-1}(1-\epsilon), \tag{4}$$

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

The inequality (4) gives us the sufficient condition for a valid allocation to a SVC request. Besides, a valid allocation should only allocate VMs onto the empty slots on physical machines. Accordingly, given a VM allocation solution for a SVC request $r =< N, \mu, \sigma >$ that allocates m VMs and $N - m$ VMs in two components divided by any link L, we (1) check whether the two components have no less than m and $N - m$ empty slots in their physical machines respectively, and also (2) compute the corresponding $u_{r,L}$ and $\sigma_{r,L}^2$ using Formula (2) and (3), and then check whether such solution is valid by examining the condition (4). The allocation solution is valid only if these two constraints are both satisfied on any physical links in the network. Note that if the new arrival request r_{K+1} is a deterministic virtual cluster request, i.e., $u_{K+1,L} = \mu \min(m, N - m)$ and $\sigma_{K+1,L}^2 = 0$, we can still use the condition (4), 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.

C. VM allocation algorithm

According to the discussion above, VM allocation in the tree topology is to allocate a 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.



(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 occupancy costs.

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 cost which 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 cost. To illustrate the problem, Fig. 3 gives an example, with a simple tree topology T in which a switch connects two physical machines 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 without explicitly considering some other metrics of interest in the algorithm, the allocation returned by TIVC algorithm [7] is not optimal on these metrics, 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.

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.

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 (4), 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 (4) for a valid allocation can also expressed as $O_L < 1$ for any link L.

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 a 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

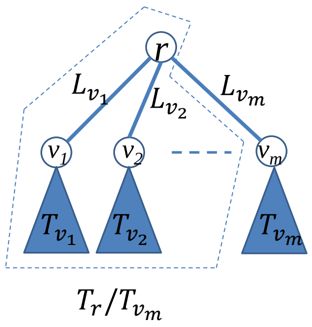


Fig. 4: The diagram for showing the optimal substructure with Lemma 2.

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 (2), (3) 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.

3) Algorithm description: Now we present our allocation algorithm. The algorithm traverses the topology tree starting at the leaves (physical machines 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 cost. 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 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 physical machine. 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.

Algorithm 1: VM Allocation Algorithm

```
Input: Datacenter tree topology $T$, SVC request $r = < N, \mu, \sigma >$,
       bandwidth reservation information of each link including
       the mean and variance of each SVC demand on it
1  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          if $l = 0$ then // leaf $v$ is a physical machine
5              $S_v[0] \leftarrow \{0, 1, \ldots, C_v\}$ ; // $C_v$ is the number of
               empty VM slots of $v$
6              for each value $h$ in $S_v[0]$ do
7                  $Opt[T_v, h] = 0$ ; // No link usage between
                   VMs in the same machine
8          else
9              $S_v[0] \leftarrow \{0\}$;
10         $T_v[0] \leftarrow v$;
11         for $i$ from 1 to $m$ do
12             $S_v[i] \leftarrow \{0\}$;
13             $T_v[i] \leftarrow T_v[i-1] \bigoplus T_{v_i}$ ;
14             for each value $e$ in $v$'s $i$-th child $v_i$'s allocable VM
               set $M_{v_i}$ do
15                 for each value $h$ in $S_v[i-1]$ do
16                     if $e + h$ is not in $S_v[i]$ then
17                         $min_v[i, e+h] \leftarrow \infty$;
18                         $S_v[i] \leftarrow S_v[i] \cup \{e+h\}$;
19                     if $i = 1$ then
20                         $Opt[T_v[i], e+h] \leftarrow$
                           $\max\left\{Opt[T_{v_i}, e+h], O_{L_{v_i}}(N, e+h)\right\}$
21                     else
22                         $Opt[T_v[i], e+h] \leftarrow$
                           $\max\left\{Opt[T_v[i-1], h], Opt[T_{v_i}, e], O_{L_{v_i}}(N, e)\right\}$
23                     if $Opt[T_v[i], e+h] < min_v[i, e+h]$ then
24                         $D_v[i, e+h] \leftarrow e$;
25                         $min_v[i, e+h] \leftarrow Opt[T_v[i], e+h]$;
26         $M_v \leftarrow \emptyset$ ;
27         for each value $h$ in $S_v[m]$ do
28             if $O_{L_v}(N, h) < 1$ then // $L_v$ is the uplink of $v$
29                 $M_v = M_v \cup \{h\}$ ;
30                 $Opt[T_v, h] \leftarrow min_v[m, h]$;
31         if $N \in M_v$ then
32             Alloc $(r, v, N)$;
33             return true;
34 return false;
35 Procedure Alloc$(r, v, x)$
36     if $v$ is a machine then
37         allocate $x$ VMs in $v$;
38     else
39         for $v$'s child $i$ from $m$ to 1 do
40             Alloc$(r, v_i, D_v[i, x])$;
41             record bandwidth occupancy for $r$ on $v$'s $i$-th link;
42             $x = x - D_v[i, x]$;
```

V. VM Allocation for Heterogeneous Bandwidth Demand

In this section, we address the VM allocation problem for heterogeneous SVC $r = < N, (\mu_1, \sigma_1), (\mu_2, \sigma_2), \ldots, (\mu_N, \sigma_N) >$, which means the bandwidth demands of VMs B_i ($1 \le i \le N$) may have different probability distributions $\mathcal{N}(\mu_i, \sigma_i^2)$. The problem is also NP-hard in deterministic case [12].

A. 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 $\{B_i | 1 \le i \le N\}$ 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 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 homogenous 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 (4) to check the validity of an allocation.

B. VM Allocation Algorithms

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 way is much more costly for the heterogeneous model to find the optimal allocation than for the homogenous model. In the homogenous case, the number of possible VM subsets that can be allocated to any subtree is at most $N + 1$, since the VMs are homogenous 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, the dependence of its time complexity on the sizes of the allocable VM sets 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 the first fit (FF) algorithm. 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, starting from r's first child subtree T_{r_1}, the algorithm greedily and sequentially places VMs into the child subtrees in order of S_v. Once a VM cannot be allocated to the current subtree, r's next child subtree is tried. As a result of the first fit, each used child subtree is assigned with a substring of S_v which is disjoint with other substrings assigned to sibling subtrees. For example, suppose n VMs are sequentially allocated to $T_{r_1}, \ldots, T_{r_m}$, then

T_{r_1} has VMs $\{v_1, v_2, \ldots, v_{d_1}\}$, denoted by $< 1, d_1 >$, T_{r_i} has $< d_{i-1}+1, d_i >$ ($1 \le d_{i-1} < d_i < n$, $1 < i < m$) and the last used subtree T_{r_m} has $< d_{m-1} + 1, n >$. As we can see, it actually searches the substring of VMs that can be allocated to the subtrees in a greedy way. The FF heuristic ensures that if an allocation solution is returned, it must be valid. However, it either may not find a solution or find suboptimal one because of its greedy strategy. To improve the chance to find a solution and its optimality, our algorithm sequentially places VMs with the above dynamic programming (DP) strategy instead of greedy strategy.

Given a sequence of N VMs, $S_N = \{1, 2, \ldots, N\}$, $< a, b >$ ($a \le b$) 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 is conducted as follows:

(1) For each physical machine 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 V-A, given that L_v divides N into two VM sets $< a, b >$ and $S_N \backslash < a, b >$;

(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, $a \le k \le b$, if $< a, k-1 >$ is in $S_v[i-1]$ and $< k, b >$ is in v's i-th child v_i's allocable VM set M_{v_i}, 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 to maximum bandwidth occupancy ratios over all possible k that has $< a, k-1 >\in S_v[i-1]$ and $< k, b >\in M_{v_i}$.

Our algorithm has time complexity $O(|V|\Delta N^4)$. With considering stochastic bandwidth demands, N VMs can be ordered by 95th percentile of their bandwidth demands. Note that the allocation algorithm in [12] which specifically aims at deterministic heterogeneous model cannot be used for our stochastic model. Although it also involves the dividing of a VM sequence, the purpose is to divide 2^N subsets into N^2 groups in this way, and examine the allocatability of all subsets in every group. Our algorithm is more efficient with considering the allocatability of only N^2 substrings and returns an optimized first fit allocation.

VI. 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.

A. 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 $< N, B >$ 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.

B. 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.

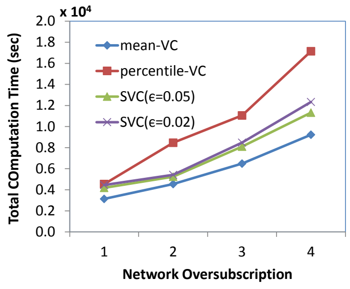


Fig. 5: Completion time with varying oversub.

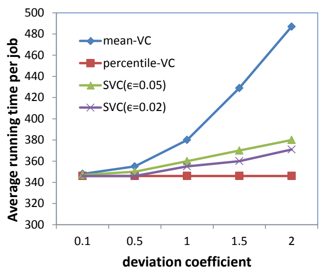


Fig. 6: Average completion time per job with varying deviation coefficient.

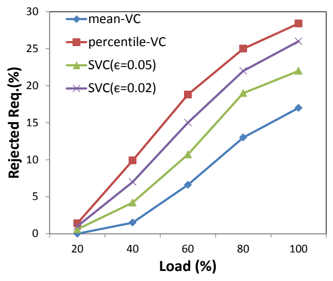


Fig. 7: Percentage of rejected requests with varying datacenter load.

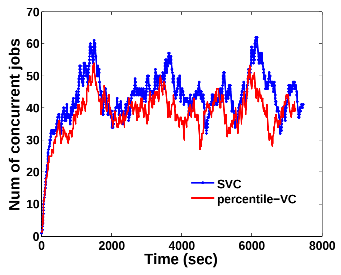


Fig. 8: Concurrent jobs for 60% load.

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. 5 shows the total completion time of 500 jobs in a batch. Fig. 6 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.6. 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.5 and 6, 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.

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. 7, 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 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. 8 show that SVC consistently achieves about 10% higher job concurrency than percentile-VC.

3) Allocation Algorithms: We compare our allocation algorithms for homogeneous SVC models with an adapted TIVC algorithm [7]. We adapt the TIVC algorithm [7] by replacing the condition of valid allocation by ours shown in (4) for computing the allocable VM sets, and additionally maintaining the information of the stochastic bandwidth demands of every

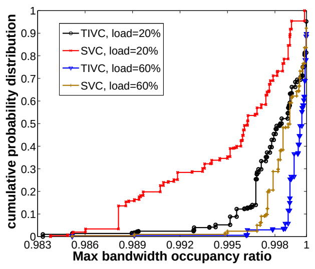


Fig. 9: The Cumulative probability distribution of sampled max bandwidth occupancy ratio.

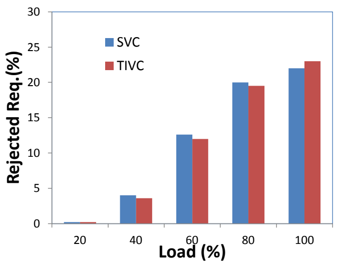


Fig. 10: Percentage of rejected requests with varying datacenter load.

SVC allocation on each link (e.g., the means and variances of $B_1^L, \ldots, B_K^L$ in Fig. 2), so it can be used for SVC allocation. However, the adapted TIVC algorithm does not involves the optimization on bandwidth occupancy cost. By the comparisons, we show the benefit of the optimization of bandwidth occupancy cost addressed by our SVC algorithms.

To evaluate the performance of algorithms on the link bandwidth occupancy cost, we assume the cloud scenario for dynamically arriving jobs as above. We sampled the maximum of bandwidth occupancy ratios among all links every time when a new job arrives. Fig. 9 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 algorithm achieves better bandwidth occupancy overhead than TIVC under both loads. Under load 20%, among all the maximum bandwidth occupancy ratios, SVC has 50% samples less than 0.996 but TIVC has only about 10%. When the load increases to 60%, the link bandwidth occupancy cost becomes higher for both algorithms. SVC and TIVC have 80% and 95% of maximum bandwidth occupancy ratios distributed on [0.997,1], respectively.

We further evaluate the request rejection rates of SVC and adapted TIVC under different loads and show the result in Fig. 10. We note that SVC and TIVC have almost the same rejection rates, which means the optimization of link bandwidth occupancy cost in SVC affects little on the optimization goal of TIVC to maximize the ability to accommodate future tenant requests. We also compared our heterogeneous SVC allocation algorithm with the first-fit algorithm in the same way. We omit the details because the results show the similar relationship between SVC and first-fit for the distribution of maximum bandwidth occupancy ratio and request rejection rates as in Fig. 9 and 10. With the optimization on the bandwidth occupancy cost, heterogeneous SVC algorithm achieves better bandwidth occupancy overhead and similar rejection rates compared with the first-fit algorithm.

VII. Conclusion

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 cloud network sharing 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. For simplicity, we assume normal distribution for the bandwidth demand in this paper, but SVC can straightforwardly use other types of probability distributions. Our future work 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 IIS-1354123, CNS-1254006, CNS-1249603, CNS-1049947, CNS-0917056 and CNS-1025652, Microsoft Research Faculty Fellowship 8300751.

References

[1] T. Gunarathne, T.-L. Wu, J. Qiu, and G. Fox, "Mapreduce in the clouds for science," in *Proc. of IEEE CloudCom*, 2010, pp. 565–572.

[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*, 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] 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.

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

[10] 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*,2012, pp. 187–198.

[11] T. Lam and G. Varghese, "Netshare: Virtualizing data center networks across services," University of California, Tech. Rep. CS2010-0957, 2010.

[12] 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.

[13] 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.

[14] 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.

[15] 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.