A Churn-prevented Bandwidth Allocation Algorithm for Dynamic Demands In IaaS Cloud

Jilei Yang, Hui Xie and Jiayun Li
Department of Computer Science and Technology, Tsinghua University, Beijing, P.R. China
Tsinghua National Laboratory for Information Science and Technology, Beijing,China
{yangjl12,jie-h12,lijiayun12}@mails.tsinghua.edu.cn

ABSTRACT

In current IaaS datacnters, the network bandwidth resource is shared across tenants in a best-effort manner. In this way, tenants are interfered by each other, which leads to unpredictable bandwidth performance and further does harm to the applications deployed in data centers. Besides, with time-varying amount of users to support, tenants choose to adapt to dynamic demands by renting different number of virtual machines(VMs) from IaaS providers at different service stage. Tenants' dynamic demand for VMs triggers the bandwidth reallocation, which produces bandwidth churn and also contributes the unpredictable performance. Those proposed strategies all put emphasis on VM's performance isolation by reserving basic bandwidth for VMs. To our best of knowledge, there is no one that has taken the tenants' dynamic demands for VMs into consideration. In this paper, we model this bandwidth allocation as an optimization problem, which is subject to the basic bandwidth guarantee and bandwidth churn. With the help of the definition of satisfaction degree for host machine, we design a simple and efficient algorithm to solve this optimization problem by using the inequality of arithmetic and geometric means. Under some experiments, we demonstrates our bandwidth allocation strategy can provide not only the predictable performance but also limit bandwidth churn, both of which improve performance of applications deployed in IaaS cloud.

KEYWORDS

Bandwidth churn, Bandwidth allocation, Network performance, Bandwidth demand

1 INTRODUCTION

As a paradigm of utility computing, cloud computing has attracted many enterprises (e.g. Dropbox and Facebook video storage) to deploy their business into cloud due to its economic resources. Using multiplexing and virtualization, the Infrastructure as a Service (IaaS) providers (e.g., Amazon EC2[1]) lease the resources in the form of virtual machines (VMs), which are charged in a pay-as-you-go price model. In this way, tenants are able to achieve isolated performance for CPU and memory. However, to the best of our knowledge, current IaaS providers do not offer performance guarantee for network bandwidth. Unlike the CPU and memory, network bandwidth in IaaS cloud is shared across different tenants in a best-effort manner, which gives rise to interference among tenants. The network performance can be influenced significantly owing to other tenants' dynamic demands when there is no basic bandwidth guarantee.

For the public cloud, the demands of tenants varies at any time, which makes the datacenter traffic highly different from any other network traffic to a large extent. Now we explain the tenants' dynamic demands from two aspect. First, some enterprises, namely tenants for the IaaS providers, choose to dynamically increase(decrease) their demands for VMs when their services need to host more(less) users. The time-varying demand for VMs can adapt to the changing amount of user, which is

also the chief reason that tenants(enterprises) are willing to migrate their service to the clouds. That means there is no need to invest the infrastructure for tenants when there is a rapid increase in user count because the invest will become an idle asset later when the surge of demand passes. From a smaller granularity, the traffic of VM is also changing over the time owing to the transformation of communication patterns. Both of the two sides depict the entire dynamic demands of tenants.

Based on the mentioned dynamic demand previously, the bandwidth sharing must handle the following two problems: 1) *No bandwidth guarantee*. Lack of necessary bandwidth isolation resembled in the CPU and memory, the network performance is interfered by each other, leading to the unpredictable execution time of jobs, further increases the risk of revenue loss for tenants and IaaS providers. 2) *Bandwidth churn*. The bandwidth reallocation triggered by the tenants' time-varying demand for VMs results in bandwidth churn problem. When tenant's demand for VMs changes, VMs will be placed on some host machines or removed. In this situation, the allocated bandwidth of VMs that still stay on host machine will fluctuate frequently due to the VMs' coming and going.

To achieve predictable performance, many bandwidth allocation schemes have been proposed to guarantee minimum bandwidth by reserving the basic bandwidth for the tenants/VMs. However, they pay no attention to the bandwidth churn problem caused by the reallocation when VMs join or leave host machines. Both of the two facets contribute to unpredictable performance that is harmful for the tenants and should be resolved appropriately.

In this paper, we model the bandwidth allocation as an optimization problem for satisfaction degree, which is subject to the basic bandwidth guarantee and limited bandwidth churn extent. To achieve the optimization objective, we try to make the different VMs on the same host machine allocated network bandwidth as equal as possible, which can be proved to be correct, on the basis of guaranteeing basic bandwidth and controlling the extent of bandwidth churn. Under the experiments, we demonstrates that our scheme can guarantee the basic network bandwidth and efficiently control the extent of bandwidth churn between two consecutive bandwidth allocations.

The remainder of the paper is organized as follows. Section II discusses the background and related work. Section III describes our model and design to resolve the above problems. Section IV evaluates the bandwidth allocation algorithm by simulations. Finally, we conclude the paper in V.

2 RELATED WORK

Recently, several mechanisms have been proposed to achieve predictable network performance. In general, there are two aspects in those works: the first is to provide minimum bandwidth guarantee, and the other is to share the network bandwidth resource inproportion

As mentioned above, Oktopus[2] and Secondnet[3] focus on predictable network performance with the thought of providing minimum bandwidth guarantee by reserving specific bandwidth for VMs. They use the static reservations across the data center to guarantee the bandwidth allocation of the entire inter-VM network. The proposed virtual topologies in those researches can provide strict network isolation at VM level. However, they can't realize high utilization for bandwidth owing to neglect of the dynamic feature of datacenter traffic. GateKeeper[4] is work conservation based on hose model and provides minimum bandwidth guarantee for VMs by shaping the traffic of VMs and the residual bandwidth is allocated in a best-effort.

The other existing policies provide network isolation at VM level with the help of proportional bandwidth provisioning. Seawall[5] is a hypervisor-based mechanism to share network proportionally on congested link according to the assigned weightsof VMs that are communicating on that link. As a statical multiplexing mechanism, Netshare[6] uses weighted fair queueing for bandwidth allocation across different tenants to achieve proportional bandwidth sharing. The policies presented in the above require to support of per-VM queue in switches for rate control, which means hard to be scaled. Meanwhile, tenants and VMs in the datacenter come and go dynamically, which gives rise to a high rate of churn. Not taking into consideration of the dynamic demand exacerbates network performance in bandwidth allocation. *Guo*[7] puts emphasis on the dynamic nature of datacenter traffic of VM, which can address one aspect of tenants' dynamic demand by using a Logistic model. Nevertheless, the case of variable tenants' demand for VMs is ignored.

With a view to the basic requirements presented in FairCloud[8], we also address the dynamic demand for coming-and-going VMs on host machines. When VMs join or leave host machines , the bandwidth reallocation will be triggered. Aiming to achieve good network performance, we set a parameter β to limit the bandwidth churn extent between two continuous allocations. Meanwhile we define *satisfaction degree* as a measurement for the bandwidth allocation state of host machine. By allocating bandwidth to VMs hosted on the same machine as equal as possible, we can make host machine achieve the best satisfaction degree. At the same time, the minimum bandwidth allocation can be guaranteed and bandwidth churn caused by continuous reallocation is also handled efficiently.

3 MODEL AND PROBLATION

Firstly, we depict the assumption and model of data center network in this part, then propose the optimization model based satisfaction degree for handling the bandwidth allocation problem.

3.1 Data Center Network Model

Based on recent works on sharing data center networks, we abstract the connections of VMs into a hose model[9] shown in Fig. 1. In this model, all VMs of tenants are connected to a simple logical non-blocking switch with guaranteed bandwidth on each access link, which means that the network bandwidth throughput of each VM can only be blocked by its access control.

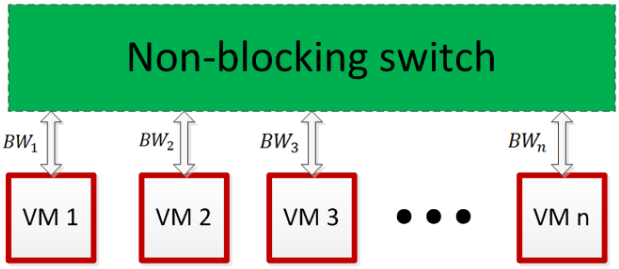


Fig. 1: Hose model, where N VMs are connected to a unblocked switch

Under the help of existing advanced researches on multi-tree topologies[9][10]and multi-path routing[11], the full bisection bandwidth has been improved considerably, which makes the physical bandwidth of host machines can be fully utilized without considering the bottleneck links inside the datacenter. In this way, the problem of managing tenant network bandwidth in the datacenter is converted to a more tractable problem of managing the host machine's network access, namely, shifting the network bottleneck from network fabric to the endpoint links that connect each host machine to the network fabric.

In datacenters, VMs are placed on some host machines according to the demands for CPU, Memory and Storage, using "pay-as-you-go"

price model. After VMs' placement, network bandwidth is allocated in "best-effort" way, which can't guarantee network performance in current IaaS datacenters. We consider the IaaS data centers consisting of *M* physical machines, $\mathcal{M}$=$\{p_1,\ p_2 \ldots p_m\}$ and every physical machine contains N VMs, $\mathcal{N}$=$\{v_1,\ v_2,\ \ldots,\ v_n\}$. With the help of hose model, we just need to control the bandwidth access to the IaaS data center on each host physical machine without considering the inside fabric. Under this circumstance, we should know the straightforward bandwidth demand of each VM. Then the bandwidth demand of VM *N* can be denoted as d_n, so all the bandwidth demand for VMs on host machine $i,\ i \in \mathcal{M}$, can be depicted as a vector, such as $D_i = \{d_1, d_2 \ldots d_n\}$.

Now, we can describe each VM by using the specific bandwidth demand. For the bandwidth allocation, we should firstly consider the guarantee for the network performance. We choose to assign a minimum bandwidth guarantee to each VM firstly as in [12]. In this way, we can achieve a network bandwidth guaranteed performance. At the same time, the specific bandwidth demand should also be assigned to each VM. With the description above, each VM can be depicted from two aspects including the minimum bandwidth guarantee and its bandwidth demand, so the VM i can be depicted as a two-dimension vector, such as $V_i = \{l_i, d_i\}$. Besides, we also need to record the rate of bandwidth allocation information for each VM. Then another parameter r_i for VM $i$is used to keep track of the allocated bandwidth. As a result, VM i can be described as a three-tuple $[l_i, d_i, r_i]$.

For the host machine, we represent server p_m using a two-tuple. First, we should figure out the total network bandwidth of the host machine as C_m. Then the residual bandwidth of server p_m is recorded as R_m after each bandwidth allocation. Asa result, the host machine can also be depicted from two aspects including the total capacity and the residual bandwidth, such as $p_m = \{C_m, R_m\}$ for the host machine p_m.

To achieve a good network performance, we use a parameter S to measure the quality of network performance for each VM. For each VM, its bandwidth demand has been proposed ahead. According to its minimum bandwidth guarantee l_i and total bandwidth demand d_i, VM i will be placed on a proper host machine with the allocated bandwidth capacity r_i. Then the parameter S can be denoted as $\frac{r_i}{d_i}$ and this parameter is called as *satisfaction degree*. In other words, the *satisfaction degree* is used to describe the state of the bandwidth allocation for VM i, namely $S_i = \frac{r_i}{d_i}(r_i \le d_i)$, which means that if VM i is allocated more network bandwidth for the demand d_i, the network performance is better and thesatisfaction degree is higher. In our paper, we focus on achieving a good bandwidth allocation state for a host machine but not for the individual VM. Then we define the satisfaction degree for the host machine p_m as the product of satisfaction degree of all the VM hosted on the same server p_m, such as

$$\prod_{i=1}^{n} \frac{r_i}{d_i}$$

,which is the objective function defined in the following optimization problem of bandwidth allocation.

Taking into consideration of the bandwidth demand of VM, we can't place VMs on any host machine arbitrarily. We assume the bandwidth allocation is r_i for the VM i hosted on server p_m when the VM i $[l_i, d_i, 0]$ has a minimum bandwidth guarantee l_i and bandwidth demand d_i. Obviously, the allocation should be bounded by the total bandwidth capacity of host machine p_m, i.e.,

$$\sum_{i=1}^{i=n} r_i \le C_m$$

In the meanwhile, the VM should be guaranteed the minimum bandwidth for network performance, so we use the constraint

$$r_i \ge L_i$$

to ensure that each VM can achieve predictable network performance, and

$$L_i = min\{l_i, d_i\}$$

Correspondingly, the allocated bandwidth should also not surpass the bandwidth demand d_i, then

$$r_i \le d_i$$

However, each bandwidth reallocation can trigger the bandwidth churn problem because of lacking of limitation for the extent of bandwidth. Our allocation strategy focuses on decreasing the bandwidth churn between two continuous allocation. We define another parameter $\beta\,(0 \le \beta \le 1)$, which is used to control the extent of bandwidth churn. For example, r_i' is the current bandwidth allocation and r_i'' is the bandwidth reallocation after a new VM joins the same host machine. Obviously, $r_i' \ge r_i''$. To control the extent of bandwidth churn, we should set the constraint as

$$r_i'' \ge \beta r_i'$$

, which means that the reallocated bandwidth r_i'' will be no less than the previous bandwidth allocation $\beta r_i'$. Using the simple constraint between two consecutive allocation, the extent of bandwidth churn can be controlled efficiently by adjusting the parameter β and the predictable network performance is also obtained further.

Now we describe a detailed scene to help understand the modeling problem. When tenant wants to increase its VM demand, the new VM $[r_i,\ d_i,\ 0]$ will join in a host machine. The bandwidth of existed VMs hosted on the same host machine will be reallocated owing to the newly joined VM and we record their previous bandwidth allocation as r_j', $j \in \{1,2,\dots,n\}$. Based on the description above, we can model the bandwidth allocation as the following optimization problem:

$$max \qquad \prod_{i=1}^{i=n} \frac{r_i}{d_i} \tag{1}$$

$$s.t. \qquad \sum_{i=1}^{n} r_i \le C_m \tag{2}$$

$$r_i \ge L_i \tag{3}$$

$$r_i \le d_i \tag{4}$$

$$r_i \ge \beta r_i' \tag{5}$$

According to (2), the sum of the allocated bandwidth should be no more than the capacity of the host machine. If the sum is less than the capacity, namely $\sum_{i=1}^{n} r_i < C_m$, it means that there is some residual bandwidth for the host machine afterbandwidth allocation and all the bandwidth demand of VMs is less than the capacity of host machine. As a result, the VM i will get all its bandwidth demand $r_i = d_i$ and its satisfaction degree is 1. This situation is a special case. However, we should process the case that the total bandwidth demand of all VMs is greater than the capacity of host machine. As a result, $\sum_{i=1}^{n} r_i = C_m$ and the allocated bandwidth r_i is not equal to the bandwidth demand, namely $r_i \ne d_i$, which means the total bandwidth demand is beyond the capacity of host machine and some VMs can't obtain the whole demand bandwidth.

3.2 Algorithm Design

According to the model analysis above, we can see that the objective function is to achieve the best satisfaction degree. In the modeling, the bandwidth demands of different VMs have been pointed out clearly, which means that the parameter d_i in the objective function (1) is a

known condition and we should obtain the bandwidth allocation r_i. Then, the problem can be converted to the optimization problem that tries to obtain the maximum value of the product r_i, namely $max \prod r_i$, where $\sum_{i=1}^{n} r_i = C_m$.

In the optimization problem, we can see that the sum of the allocated bandwidth r_i is a fixed value and we try to achieve the max value of their product. Based on the analysis above, we can design our algorithm inspired by the nature of *AM-GM inequality* described as follows:

$$\frac{x_1 + x_2 + \cdots + x_n}{n} \geq \sqrt[n]{x_1 \cdot x_2 \cdots x_n}$$

and that equality holds if and only if $x_1 = x_2 \cdots = x_n$. The condition that the *AM-GM inequality* satisfies the equal case indicates that the closer the $x_i (i \in 1,2 \cdots n)$ is, the larger the value of their product is. Then we can conclude the following equality:

Theorem $\forall x_i > 0, i \in \{1,2,\cdots,n\}, \sum_{i=1}^{n} x_i = C$. If x_i is closer to each other, that's to say, the variance is smaller, $\prod_{i=1}^{n} x_i$ is larger.

In this section, we will give a detailed description about our allocation algorithm design. In the bandwidth allocation system as in Fig. 2, the host machine has the total system information, including its own bandwidth information about its total bandwidth capacity C, the residual bandwidth capacity R and the allocated bandwidth information of VMs on the host machine. As for the VM i on host machine, it can be depicted as a three-tuple $[l_i, d_i, r_i]$, which is recorded in bandwidth allocator of host machine. Based on the analysis above, we can make the bandwidth allocation for each VM as even as possible to achieve the best satisfaction degree for the host machine. However, the bandwidth demand of VMs is different and so is the minimum bandwidth guarantee, which impedes the idea that we allocate the bandwidth to VMs equally. Aiming

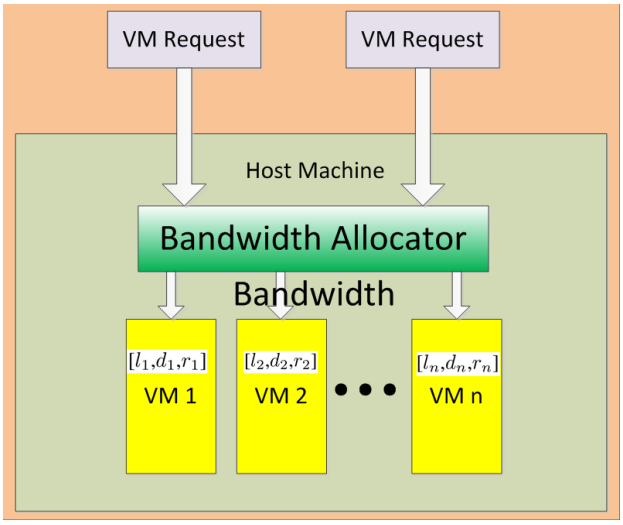


Fig. 2:Bandwidth allocation in host machine

at to solve this problem, we propose a multi-stage algorithm that can make the bandwidth allocation as equal as possible under the restriction above.

When some VMs join or leave a host machine, the bandwidth reallocation can be triggered. At first, we should update the bandwidth allocation information as follows. As for the VMs hosted on physical machine, their original record information is a three-tuple $[l_i,d_i,r_i]$, then we update $l_i = max\{\beta r_i, l_i\}$ and $r_i = l_i$. In the case that there are some new VMs to join host machine, along with the new VM request $[l_j,d_j,0]$, all the three-tuple of VMs $[l_i,d_i,r_i]$ will be sorted by the third dimension r_i. In the meanwhile, we decide whether the bandwidth requests can be accepted in the allocation using the following criteria:

$$result = \begin{cases} accepted, & \sum_{i=1}^{i=n} l_i < C \\ rejected, & \text{otherwise} \end{cases}$$

If the request is accepted, we can divide the case into two kinds of situations. Firstly, when the total bandwidth demand is less than the capacity of host machine, namely $\sum_{i=1}^{n} d_i \leq C$, all the bandwidth demand of VMs on the same host machine can be satisfied, so $r_i = d_i$. The other case is that the capacity of the host

machine can't satisfy all the bandwidth demand of the VMs, then we call our bandwidth allocation algorithm..

To achieve network performance, the minimum bandwidth should be guaranteed. In other words, the minimum bandwidth should be allocated firstly. Inspired by *Theorem 1*, we allocate network bandwidth to VMs as equally as possible under the premise of guaranteeing network performance. After VMs' bandwidth allocation information preprocessing and sort in a ascending order as described above, we choose to decide how much the amount of the bandwidth is allocated to VMs. In this processing, we will determine two parameters, one is the minimum bandwidth demand of the VMs that have the least bandwidth guarantee value. Owing to the bandwidth demand sort, we just need to search the VMs that have the minimum bandwidth guarantee in the set A, where all VMs wait to be allocated. Then the value of minimum bandwidth demand can be defined as follows:

$$MinDnd = \{d_i | l_i = l_0, i \in A\}_{min}$$

For example, [20 , 30 , 20], [20 , 37 , 20], [20 , 32 , 20], [35 , 40 , 35] are the sorted bandwidth demand. In the example, we can see that the minimum bandwidth guarantee is 20 and there are three VMs meeting the condition that they have the minimum bandwidth guarantee. Among the three VMs bandwidth request [20,30,20], [20,37,20], [20,32,20], their bandwidth demands are 30 , 37 , 32 respectively. Then the least bandwidth demand is 30 , namely $MinDnd = 30$. The other parameter is the second least bandwidth guarantee if it exists, namely $SecGua = 35$ and the least bandwidth guarantee is 20. According to the derived two parameters $MinDnd$ and $SecGua$, we elicit the next bandwidth value allocated in the next allocation round using the following rule:

$$UpBand = \begin{cases} MinDnd & , \text{ If SecGua} = 0 \\ \min\{MinDnd, SecGua\}, & \text{otherwise} \end{cases}$$

,which gives the upper-limit to the next bandwidth allocation round. As listed in the

Algorithm 1: Bandwidth Allocation Algorithm

Input: New Bandwidth demand vector $[l_j, d_j, 0]$;
Allocated bandwidth vector$[l_i, d_i, r_i]$;
Total bandwidth capacity of host machine: C;
Residual bandwidth of host machine: R;
Parameter of bandwidth churn control: β;
The bandwidth allocation of current stage: $CurBand$;
Set of VMs waiting to bandwidth allocation: A;
Set of VMs that finish bandwidth allocation: B;
Output: Allocated bandwidth vector
$[l_i, d_i, r_i], i \in \{1, 2, \cdots, N\}$
$l_i = max\{\beta r_i, l_i\}$, $r_i = l_i, i \in 1, 2, \cdots, N$;
$CurBand = l_0$;
Sort all the bandwidth demand vector by r_i in ascending;
Move all the VM to the set A;
while *set A is not empty* $\vee$ $R \neq 0$ **do**
  $MinDnd = min\{d_i | l_i = l_0, i \in 1, 2, \cdots, N\}$;
  $SecGua = \{l_i | l_i \leq l_j \forall l_j \neq l_0, i, j \in 1, 2, \cdots, N\}$;
  if *SecGua exists* **then**
    $UpBand = MinDnd$;
  else
    $UpBand = min\{MinDnd, SecGua\}$
  Bandwidth allocated to VM is
  $BandAlloc = min\{\frac{R}{M}, UpBand - CurBand\}$;
  $r_i = r_i + BandAlloc, i \in \{j | l_j = l_0, j \in 1, 2, \cdots, N\}$
  if $r_i = d_i$ **then**
    remove VM i from set A to set B;
  $CurBand = CurBand + r_i$;
  $R = R - M * BandAlloc$;
return $[l_i, d_i, r_i], i \in 1, 2, \cdots, N$;

example, the current allocated bandwidth is their guaranteed bandwidth for all the VMs. If allocated continually, the bandwidth of wait-allocated VMs that have the leastallocated bandwidth can reach to the upper-limit $UpperBandwidth$. To allocate bandwidth equally, the residual bandwidth R of host machine is firstly allocated among the amount M of VMs that have the least allocated bandwidth currently. Then the allocated bandwidth in this round is

$$BandAlloc = min\{\frac{R}{M}, UpperBand - CurBand\}$$

Then the VMs that have the least allocated bandwidth obtain bandwidth from host machine and update their bandwidth allocation as follows:

$$r_i = r_i + BandAlloc$$

Then the residual bandwidth of host machine is also updated

$$R = R - M * BandAlloc$$

If there are someVMs satisfying that their allocated bandwidth is equal to the bandwidth demand, namely $r_i = d_i$, their bandwidth allocation will be finished completely. If all the VMs have finished bandwidth allocation or host machine has no residual bandwidth, the bandwidth allocation algorithm is terminated.

3.3 Algorithm Analysis

From the description about our algorithm, we should firstly sort the bandwidth demand vector, which uses the time of $O(nlogn)$. In the bandwidth allocation process, the amount of the next round bandwidth allocation is determined by traversing VMs demands vector and this step just costs $O(1)$ with the help of the sort above. Based on the analysis above, the bandwidth allocation will be executed many times, which is determined by the total amount of the different VM's bandwidth guarantee, demand and allocated, namely $3 * O(n)$. Then the total the time complexity of our bandwidth allocation algorithm is $O(nlogn) + O(3n)$, namely $O(nlogn)$. The bandwidth allocation based on game-theory[?] develops a centralized allocation method and a distributedone. The first is $O(n^2)$ and the rate of convergence of distributed allocation is determined by its step-size ξ. Compared with other bandwidth allocation scheme, we can just limit the allocation in the host machine and the time complexity is determined only by the amount of the hosted VMs. In general, the amount of the VMs in a host machine is the order of ten, which is very small for the thousands of the machines in IaaS data centers.

4 EXPERIMENT

To validate the effectiveness of our proposed scheme, we simulate an IaaS data center with homogeneous servers, of each equipped with network bandwidth capacity of 1Gbps. As assumed in hose-model, the data center has a full-bisection bandwidth networkeref[8] and we allocate bandwidth using Traffic Control(TC) tool to limit the rate of each VM in a linux based OS.

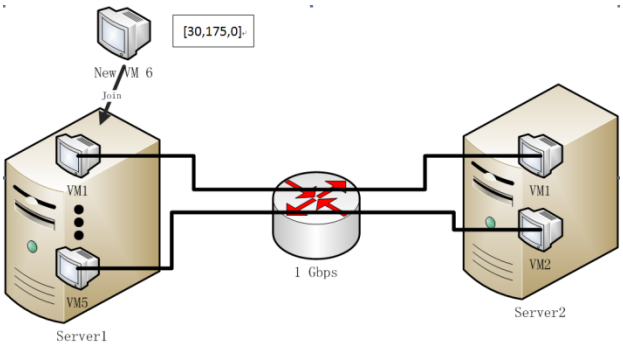


Fig.3:Experiment environment

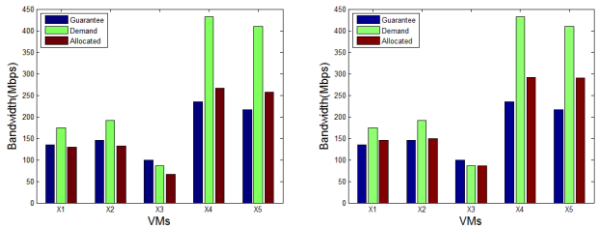
Table I:Current bandwidth allocation information

VM	Guarantee(Mbps)	Demand(Mbps)	Allocated(Mbps)
X1	135	175	153
X2	145	192	161
X3	100	86	76
X4	235	432	308
X5	217	410	305

As pictured in Fig. 3, two host machines are linked through a switch with capacity of 1 Gbps. The tenant *A* has five VMs placed on the server 1 and each VM communicates with two other VMs hosted on the server 2. The current detailed bandwidth demandand allocated bandwidth information of tenant A's VMs are listed in Table. I. As mentioned in Sec. I, the bandwidth in data center is allocated in a best-effort way. In this way the bandwidth allocation will be interfered each other when there are some VMs to join the same host machine.

Owing to the tenants' dynamic demand for VMs, a new VM request arrives at the host machine that host all the VMs of tenant A. The new VM has a bandwidth demand of 175Mbps and its guaranteed bandwidth is 30Mbps. Based on the best-effort bandwidth allocation, the bandwidth of all the existed VMs on the same host machine will be decreased owing to the bandwidth sharing across the tenants in the data center. As illustrated in Fig. 4(a), VMs(X1-X3)

get less bandwidth allocation than their basic bandwidth guarantee, so they are not guaranteed basic bandwidth in reallocation with best-effort way, while our designed algorithm achieves a guaranteed network performance as shown in Fig. 4(b). As a contrast in Fig. 5, our multi-stage allocation algorithm can lower the bandwidth churn extent effectively from 15% to 5% by using a tunable parameter β and all of VMs achieve their network performance.



(a) Bandwidth allocation in best-effort way.

(b) Bandwidth allocation in multi-stage algorithm.

Fig.4:Bandwidth Allocation

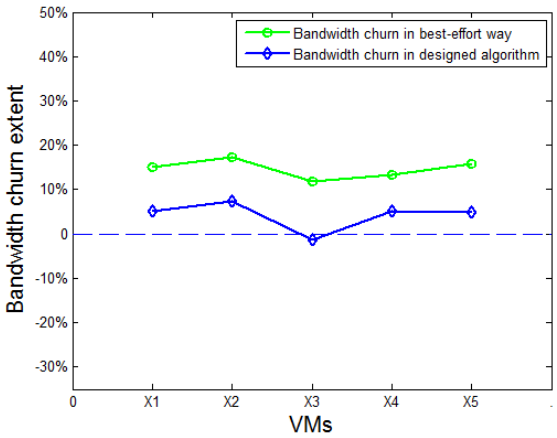


Fig.5:Extent of bandwidth churn caused by dynamic demand

In our experiment, we can see that VMs that have less bandwidth demand have a higher satisfaction degree than those with more bandwidth demand. That means our bandwidth allocation has a preference for the VMs with little bandwidth demand.

5 CONLUSION AND FUTURE WORK

In this paper, we use the hose model to analysis the bandwidth allocation problem in IaaS data center. By defining the *satisfaction degree* as the measurement of the bandwidth allocation state, we model the bandwidth allocation problem as an optimization problem, whose objective is to achieve the best satisfaction degree under some restrictions. To achieve an ideal network performance, we not only guarantee the minimum bandwidth but also limit the bandwidth churn extent with the parameter β between two continuous bandwidth allocation. We designed a multi-stage allocation algorithm based on the *AM-GM inequality* to handle the bandwidth allocation in host machine. As shown in our experiment, each VM achieve guaranteed network performance successfully and the bandwidth churn of each VM is also decreased from 15% to 5%.

In our model, we must know the explicit bandwidth demand of VMs or tenants head. With the help of the work [13], we can predict some bandwidth with dynamic demand. As mentioned above, our allocation algorithm has an unfair for the VMs with large bandwidth demand. In the future, we are going to make a limitation on the upper limit of bandwidth allocation to realize the fairness across different tenants and VMs.

REFERENCES

[1] "Amazon elastic compute cloud." [Online]. Available: http://aws.amazon.com

[2] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in proc. *of ACM SIGCOMM*,2011

[3] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: a data center network virtualization architechture with bandwidth guarantees," in *proc. of ACM CoNext*, 2010.

[4] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes, "Gatekeeper: supporting bandwidth guarantees for multi-tenant datacenter networks," in *proc. of 3rd workshop on I/O virtualization*. USENIX,2011.

[5] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, "Sharing the data center nerwork," in *proc. of USENIX NSDI*, 2011.

[6] T. Lam and G. Varghese, "Netshare: virtualizing bandwidth within the cloud," UCSD, Tech. Rep., 2009.

[7] J. Guo, F. Liu, X. Huang, J. C. Lui, M. Hu, Q. Gao, and H. Jin, "On efficient bandwidth allocation for traffic variability in datacenters," in *proc. of IEEE INFOCOM*, 2014.

[8] 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.

[9] A. Greenberg, J. R.Hamilton, N. Jain, S. Kandula, and C. Kim, "Vl2: A scalable and flexible data center network," in *proc. of ACM SIGCOMM*,2009.

[10] C. Guo, G. Lu, D. Li, H. Wu, X. Z. Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: A high performance, server-centric network architecture for modular data centers," in *proc. of ACM SIGCOMM*, 2009.

[11] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipth tcp," in *proc. of ACM SIGCOMM*, 2011.

[12] J. Guo, F. Liu, D. Zeng, J. C. Lui, and H. Jin, "A cooperative game based allocation for sharing data center networks," in *proc. of IEEE INFOCOM*, 2013.

[13] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[14] J. Schad, J. Dittrich, and J.-A. Q. . e Ruiz, "Runtime measurements in the cloud: observing,analyzing, and reducing variance," in *proc. of VLDB*, 2010.