IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 4 Issue 6, June 2017
ISSN (Online) 2348 – 7968 | Impact Factor (2017) – 5.264
www.ijiset.com

Virtualization of Data Center towards Load Balancing and Multi-Tenancy

G. Srinivas Yadav

Assistant Professor, Dept of CSE, Sreenidhi Institute of Science & Technology,Hyderabad,mukundhach@gmail.com

Abstract

Virtualization is an essential step before a bare metal data center being ready for commercial usage, because it bridges the foreground interface for cloud tenants and the background resource management on underlying infrastructures. A concept at the heart of the foreground is multi-tenancy, which deals with logical isolation of shared virtual computing, storage, and network resources and provides adaptive capability for heterogeneous demands from various tenants. A crucial problem in the background is load balancing, which affects multiple issues including cost, flexibility and availability. In this work, I propose a virtualization framework that considers these two problems simultaneously. My framework takes advantage of the flourishing application of distribute virtual switch (DVS), and leverages the blooming adoption of OpenFlow protocols. First, the framework accommodates heterogeneous network communication pattern by supporting arbitrary traffic matrices among virtual machines (VMs) in virtual private clouds (VPCs).The only constraint on the network flows is the bandwidth of server's network interface. Second, our framework achieves load balancing using an elaborately designed link establishment algorithm. The algorithm takes the configurations of the bare metal data center and the dynamic network environments as inputs, and dynamically applies a globally bounded oversubscription on every link. Our framework concentrates on the fat tree architecture, which is widely used in today's data centers.

Keywords: Virtualization, load balancing, multi-tenancy, datacenter, data center networks, distributed virtual switch.

I. INTRODUCTION

Virtualization is the enabling technique to facilitate the sharing of resources in data centers, which transforms a huge collection of bare-metal hardware into cloud infrastructure with high flexibility, predictable performance, reliability, controllability, and security [1].

Virtualization of servers introduces a supervisor between the hardware and the operating systems, and produces isolated computing units known as virtual machines (VMs). Virtual machines are well studied before the appearance of data centers, dated back to 1970's [2]. On the other hand, virtualization of networks is much more delayed because the relatively static nature of networks is in stark contrast to the dynamic nature of computing units. This imbalance between virtualization of servers and virtualization of networks becomes an important driven force of software-defined networking (SDN) and its underlying distributed virtual switch (DVS).SDN is a revolutionary innovation to networks in the sense that network control is decoupled from the data forwarding function and is directly programmable. The result is an extremely dynamic, manageable, cost-effective, and adaptable architecture that gives administrators unprecedented programmability, automation, and control. The *OpenFlow* protocol is a fundamental element for building SDN solutions.

It is an industry-standard SDN communications protocol, allowing operators to address complex network behavior, optimize performance, and leverage a richer set of capabilities. OpenFlow allows direct access to and manipulation of the forwarding plane of network devices such as switches and routers, both physical and virtual (hypervisor-based). OpenFlow also provides a unified interface that aggregates and manages all the hardware switches, even when the heterogeneous switches may be from different manufacturers and using different scripting languages [3]. It allows fine-grained flow-level control of switching, which makes it possible to apply flexible routing policies without being buried by switch-by-switch flow table configurations, thus significantly reduces operations and management complexity. In addition, OpenFlow is layered over TCP, making it compatible with most network protocol stacks. A virtual switch within one server can transparently join with a virtual switch in another server (see Fig. 1), making communications of VMs more efficient.

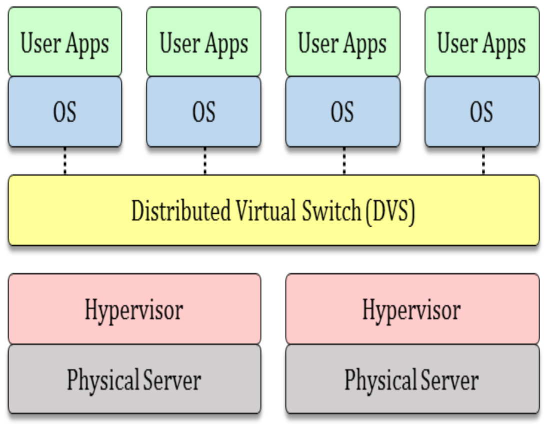

Fig.1. Illustration of distributed virtual switch(DVS)

In cloud computing, load balancing is required to distribute the dynamic local workload evenly across all the nodes and links. It helps to achieve a high user satisfaction and resource utilization ratio by ensuring an efficient and fair allocation of every piece of resource. Proper load balancing aids in minimizing resource consumption, implementing fail-over, enabling scalability, avoiding bottlenecks and over provisioning. For these reasons, load balancing lies at the heart of the back-end of network virtualization, and remains a hot topic in the research community. Much work [4] approaches this problem from different angles and, various load balancing practices in data centers have been proposed.

In a load balanced environment, traffic on the links should not change dramatically throughout the network. In this case, bandwidth demands on the links are regulated so that oversubscription is under control. Therefore, oversubscription, the ratio of the worst-case aggregate bandwidth demand on a link to the bandwidth capacity of the link, can be used as a metric to evaluate the outcome of load balancing. If a network topology allows 1:1 oversubscription on its links, then all the servers connected to the network have the potential to communicate with other servers at full bandwidth of their local network interface cards (NICs). However, due to the cost concern, most commercial data center designs adopt oversubscription larger than 1 [5]. Our virtualization framework utilizes an online algorithm to achieve an averaged over subscription throughout the data center. The algorithm is described in Section IV.

In order to accommodate the dynamic connectivity demands from tenants, my framework gives each tenant an illusion that, all the computing nodes within the private cloud, along with the virtual links interconnecting the nodes, form a complete graph. Therefore, the heterogeneous nature among the private clouds is no longer a problem, because any traffic matrix is easily realizable, as shown in Fig. 2.

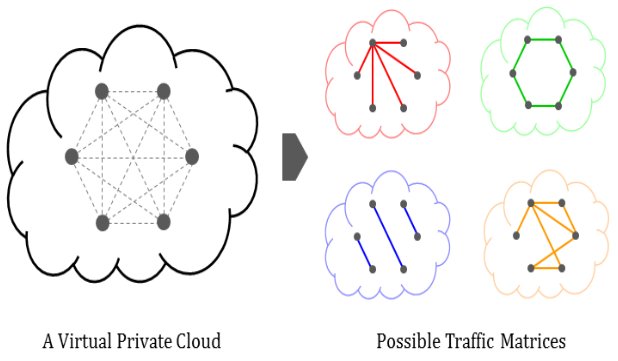

Fig 2: Our virtualization framework provides private clouds with complete graph- like connection illusion among their VMs.

For the complete graph illusion, we adopt the traffic model proposed in [6], because it is both clear enough and general enough - the only constraint is that the egress and ingress traffic being bounded by local network interface. Because of these features, this model has been commonly used in previous works on DCNs [7], [8], [9].

This framework is featured by:

- Load balancing is achieved globally in the data center. Hot spots are eliminated by an upper bounded oversubscription for every link in the data center. This upper bound is explicitly determined by the configurations and network environments of the fattree data center.

- Each virtualized private cloud is ready for any possible traffic matrix under full supports to the hose model. A VM is able to communicate with other VM(s) in the private cloud in parallel, as long as the NICs of the corresponding physical servers are not exhausted.

II. RELATED WORK

Several VM placement schemes are proposed in [10] and [11] in order to support nonblocking multicast virtual networks. However, oversubscription, the currently widely adopted feature, is prohibited in these works, which leads to luxurious usage of scarce network resource. In addition, [20] and [21] focus on the VM placement problem only, the path finding problem for the communications among VMs is omitted. In this paper, we propose a full-stack virtualization framework including both VM placement and virtual link establishment.

In [12] and [13] an online flow scheduling algorithm is proposed to perform load balancing in data centers, however, [12] is limited in the context of *non*-virtualized data centers. Also, [12] does not take the dynamic network traffic environment into consideration. The fact that virtualization has become Ubiquitous in data centers makes our framework practically valuable. Moreover, I am considers this framework is both the static configuration of underlying infrastructure and the dynamic network traffic environment, making it more adaptive.

Another related work is [14]. The network model used in [14] classifies links into over-loaded ones and under-loaded ones, according to that the link load is greater than half the capacity or not. This model is clear and straightforward, but is lacks flexibility. Also, in [14], only one-to-one communications are considered. If a large piece of data has multiple destinations (which is not uncommon in today's data centers), repeated one-to-one communication would waste both bandwidth and time. In this framework, arbitrary traffic matrices are supported among VMs in a VPC, including but not limited to one-to-one, one-to-many and one-to-all communications.

III. NETWORK MODEL

Fat-tree as a DCN architecture was firstly proposed in [15]. The fabric of a fat-tree DCN contains three tiers of switches, which are ToR switches, aggregation switches, and core switches in the bottom-up order. The ToR switches and the aggregation switches are distributed in a number of pods, as shown in Fig. 3. It is clear that there are no direct east-west traffic between different pods, therefore, the switches in each pod can be seen as a large edge switch that connects all the servers of that pod. In this way, a fat-tree DCN is abstracted into a 2-tier model, with edge switches at the bottom and core switches at the top.

The 2 tiers of switches in our model are shown in Fig. 3. Edge switches provide links for the servers to be connected to the network, and handle intra-edge traffic as well. Core switches, on the other hand, take care of inter-edge traffic. This model could be denoted as an (m, n, r) fat-tree, where m is the number of core switches, n is the number of servers connected to one edge switch, and r is the number of edge switches. Clearly, an (m, n, r) fat-tree DCN connects N = nr servers to the network.

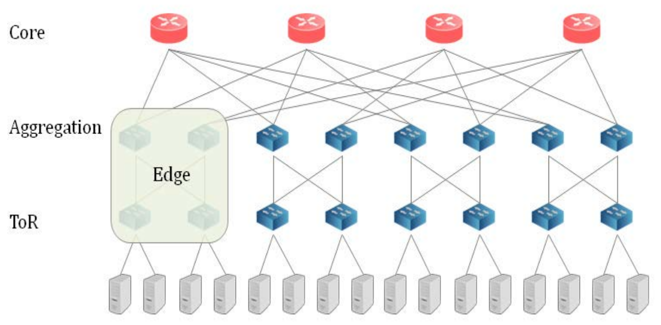

Fig 3: Abstracting a 3-tier fat-tree DCN into a 2-tier model.

There is exactly one duplex link between any pair of core switch and edge switch, and there are no links between any two switches in the same tier, forming a bipartite graph like structure. A network flow from one edge switch to another edge switch must use some core switch along with corresponding links between the core switch tier and the edge switch tier. For the sake of discussions on this link establishment algorithm, i can view a duplex link as two simplex links: an up link from the edge tier to the core tier and a down link from the core tier to the edge tier. Due to the highly volatile nature of data center traffic, myself adopt a very concise and general traffic model known as the hose traffic model [6], where each server has a maximum ingress bandwidth capacity and a maximum egress bandwidth capacity. Any traffic matrix that is consistent with the ingress/egress bandwidth capacity must be included in this model. In this framework, the ingress/egress bandwidth capacity assign is simply the bandwidth capacity of a server's network interface. In this way, take into account of the "worst-case" traffic patterns in our framework. For simplicity, myself set the ingress/egress bandwidth capacity of the each server to 1, and denote the normalized bandwidth capacity of the links between the core switch tier and the edge switch tier as c.

Here define a few frequently used notations. The first notation introduce is the *overall utilization* of the servers, denoted by u, which is defined as the ratio between the equivalent total number of used servers and N, the total number of servers in a fat-tree DCN. It describes the proportion of the used servers throughout the data center. Utilization of servers can also be local, for example, i can get similar statistics among the servers connected to any one edge switch. Therefore, use *local utilization*, ui, to describe the used proportion of all the servers connected to the ith edge switch, where i = 0, 1, 2, . . . , r−1. The number of VMs contained may vary among different VPCs. Here use s to denote the size of a VPC, which is defined by the equivalent number of physical servers that are used by all its inner VMs. Use s_{max} to denote the size of the largest VPC in a data center.

IV. VIRTUALIZATION FRAMEWORK

In this section, described proposed virtualization framework, and present detailed explanations and demonstrations. The first phase of the framework is VM placement. In response to a cloud tenant's request, a number of VMs are allocated. The VM placement phase selects from the pool of physical servers to accommodate the VMs. In this phase, the framework follows the guideline of traffic localization [18], which places VMs of a VPC in servers that topologically close to each other, in order to confine large chunks of traffic at hierarchically low level switches. For example, in this model of the fat-tree DCN, if two servers are connected to the same edge switch, then an end-to-end flow from one server to the other uses only switching power of the edge switch. On the other hand, if the two servers are connected to different edge switches, then the same flow between them has to use additional links between the two tiers and additional switching power of the core switches. Therefore, it is desirable to localize the VMs in some way, so that as fewer as possible edge switches are involved in a VPC.

The VM placement of the framework uses a greedy Strategy, which is shown as Algorithm 1. The algorithm searches for the edge switch that currently connects the largest number of free physical servers. After that, VMs are placed into the free servers, if not all VMs are placed, the algorithm continues the same search and placement until all the VMs of the VPC are placed.

```
1:  $\mathcal{V}_{placing} = \{$all VMs of the VPC$\}$;
2:  $\mathcal{V}_{placed} = \phi$;
3:  $\mathcal{E}_e = \{$all eligible edge switches$\}$;
4:  while $\mathcal{V}_{placing} \neq \phi$ and $\mathcal{E}_e \neq \phi$ do
5:      $\hat{i} = \arg\min_{i:E_i \in \mathcal{E}_e} u_i$;
6:      $\mathcal{E}_e = \mathcal{E}_e \setminus \{E_{\hat{i}}\}$;
7:      place as many as possible VMs to servers under $E_{\hat{i}}$;
8:      $\mathcal{V}_r = \{$VMs placed just now$\}$;
9:      $\mathcal{V}_{placing} = \mathcal{V}_{placing} \setminus \mathcal{V}_r$;
10:     $\mathcal{V}_{placed} = \mathcal{V}_{placed} \cup \mathcal{V}_r$;
11: end while
12: if $\mathcal{E}_e == \phi$ then
13:     return INFEASIBLE;
14: end if
15: return SUCCESS;
```

Algorithm 1: VM placement of the FrameWork

The second phase of the framework is link establishment. After the previous phase, it deals with the inter-edge traffic. The framework now selects appropriate core switches and links between the edge switch tier and the core switch tier to accommodate the inter-edge flows. In the VM placement phase, it use the static strategy [16]. That is, once computing resource of a physical machine is allocated as a VM, it is dedicated to the VPC's computing tasks during the VPC's lifetime, which avoids the performance degradation introduced by VM migrations. Unlike the first phase, it uses the dynamic strategy [16] in the second phase. To be specific, no dedicated physical links or switch hardware are allocated to a VPC. Instead, for each inter-edge flow, it assign links and switches according to an online algorithm shown in Algorithm 2.

Current method uses the dynamic strategy, because the Open- Flow/DVS combination provides a global view and control interface to all the network resources in the DCN. More importantly, by using the dynamic strategy, global optimization becomes possible and oversubscription is upper bounded for all the links throughout the data center.

For each link, it can calculate the ratio of the total bandwidth demand of all the flows being routed on the link to its bandwidth capacity. Note that it is possible for the subscription of a link to exceed 100%, which means that the total bandwidth demand of traffic flows occupying a network link may surpass the link capacity.

In this case, the network is oversubscribed and packet losses could occur. As oversubscription is widely adopted in DCNs to avoid network resource under-utilization and reduce cost [15], oversubscribed links are fairly common in DCNs. In these cases, contending flows on these congested links receive a fraction of their demanded bandwidth. For example, assume there are only two contending flows sharing one link. One of them demands 0.8 of the link capacity and the other demands 0.4, then the former receives 2/3 of the capacity and the latter receives 1/3.

V. PERFORMANCE EVALUATIONS

Here proposed an event driven simulator to validate the proposed framework. The simulator could handle events such as arrival or departure of cloud tenants, and beginning or ending of an inter-edge flow. These events occur in a randomized manner, which simulates the properties of lack of future knowledge in a network controller. In this simulation, private clouds have various traffic matrices, i.e., one-to-one, one-tomany, etc, or a hybrid of them (analogous to the scenarios in Fig. 2). It runs simulations under different fattree configurations and dynamic network environments, until hundreds of thousands of flows are handled, and statistics are collected during this process.

```
//initialize available core switches;
$\mathscr{C}_a = \phi$;
for each core switch $C_i$ do
    if $up\_demand[src][i] + w \le c \cdot O$ then
        $\mathscr{C}_a = \mathscr{C}_a \cup \{C_i\}$;
    end if
end for
//initialize candidate core switches
$\mathscr{C}_c = \phi$;
$\mathscr{E}'_{dst} = \mathscr{E}_{dst}$;
while $\mathscr{E}'_{dst} != \phi$ do
    //calculate the coverage for each available core switch;
    for each core switch $C_i \in \mathscr{C}_a$ do
        $C_i.cvrg = 0$;
        for each edge switch $E_j \in \mathscr{E}'_{dst}$ do
            if $down\_demand[i][j] + w \le c \cdot O$ then
                $C_i.cvrg = C_i.cvrg + 1$;
            end if
        end for
    end for
    //select core switch covering the most elements in $\mathscr{E}'_{dst}$
    $\hat{i} = \arg\max_{i:C_i \in \mathscr{C}_a} C_i.cvrg$;
    $\mathscr{C}_c = \mathscr{C}_c \cup \{C_{\hat{i}}\}$;
    //update available core switches
    $\mathscr{C}_a = \mathscr{C}_a \backslash \{C_{\hat{i}}\}$;
    //update $\mathscr{E}'_{dst}$
    $\mathscr{E}'_{dst} = \mathscr{E}'_{dst} \backslash \{$edge switches covered by $C_{\hat{i}}\}$;
end while
//Initialize selected core switches
$\mathscr{C}_s = \phi$;
for each edge switch $E_j \in \mathscr{E}_{dst}$ do
    //find a core switch in $\mathscr{C}_c$ to cover it
    $\hat{i} = \arg\min_{i:C_i \in \mathscr{C}_s} down\_demand[i][j]$;
    establish a down link between $C_{\hat{i}}$ and $E_j$;
    $down\_demand[i][j] = down\_demand[i][j] + w$;
    //update $\mathscr{C}_s$
    $\mathscr{C}_s = \mathscr{C}_s \cup \{C_{\hat{i}}\}$;
end for
for each core switch $C_i \in \mathscr{C}_s$ do
    establish up link between $E_{src}$ and $C_i$;
    $up\_demand[src][i] = up\_demand[src][i] + w$;
end for
```

Algorithm2 : Link Establishment for Inter edge network flow.

Simulations are run for different cases of network environments. In each case it first compute O using (17), and run simulations to find the cumulative distribution function of bandwidth demands over all the up/down links in the fat-tree. It can see from the figures that, at the back-end, the oversubscription is successfully bounded by computed values, regardless of the types of fat-tree configurations or network environments. Also, it show the results for different traffic patterns in VPCs, including one-to-one and hybrid communications, demonstrating that arbitrary traffic matrices are supported at the front-end of the data center.

VI. CONCLUSIONS

In this paper, proposed a fat-tree data center virtualization framework based on the aggregated control of all the switching elements provided by the OpenFlow/DVS combination. The framework utilizes the structural features of the fat-tree so as to place cooperating VMs in a localized manner. Also, the framework leverages the topological symmetries and rich connectivity of the fat-tree in the link establishment algorithm. The framework generates multi-tenancy oriented private clouds, offering great flexibility to cloud users. Arbitrary traffic matrices are supported in the private clouds, allowing VMs to communicate with multiple destinations in parallel under the hose traffic model. The framework achieves global load balancing on the underlying physical network, delivering predictable performance. Moreover, it is adaptive to dynamic network environments, such as server utilization and VPC size. Finally, simulation results have validated that the framework can successfully handle both the foreground and background tasks.

VII. REFERENCES

[1] N.M. Chowdhury, Mosharaf Kabir, and Raouf Boutaba, “Network virtualization: state of the art and research challenges,” IEEE Com- munications Magazine, 2009.

[2] E.W. Pugh, L.R. Johnson, and J.H. Palmer, IBM’s 360 and Early 370 Systems, MIT Press, 1991.

[3] N. McKeownck, et al, “OpenFlow: enabling innovation in campus networks,” ACM SIGCOMM Comput. Commun. Rev., vol. 38, no.2, pp. 69-74, 2008.

[4] N.J. Kansal and I. Chana, “Existing load balancing techniques in cloud computing: a systematic review,” Journal of Information Systems and Communication, vol. 3, no. 1, pp. 87-91, 2012.

[5] Cisco Data Center Infrastructure 2.5 Design Guide, [Online]. Available: http://www.cisco.com.

[6] N.G. Duffield, et al, “A flexible model for resource management in virtual private networks,” ACM SIGCOMM, 1999.

[7] A. Greenberg, et al, “VL2: a scalable and flexible data center network,” ACM SIGCOMM Comput. Commun. Rev., vol. 39, no. 4,2009.

[8] A.R. Curtis, S. Keshav, and A. Lopez-Ortiz, “LEGUP: Using heterogeneity to reduce the cost of data center network upgrades,” ACM CoNext, 2010.

[9] N.G. Duffield, et al, “Resource management with hoses: point-tocloud services for virtual private networks,” *Networking, IEEE/ACM Transactions on*, vol. 10, no. 5, pp. 679-692, 2002.

[10] J. Duan, Z. Guo, and Y. Yang, “Cost efficient and performance guaranteed virtual network embedding in multicast fat-tree DCNs,” *Computer Communications (INFOCOM), 2015 IEEE Conference on*,2015.

[11] J. Duan, Z. Guo, and Y. Yang, “Embedding nonblocking multicast virtual networks in fat-tree data centers,” *Parallel & Distributed Pro- cessing (IPDPS), IEEE International Symposium on*, 2015.

[12] Z. Guo, J. Duan, and Y. Yang, “On-line multicast scheduling with bounded congestion in fat-tree data center networks,” *Selected Areas in Communications, IEEE Journal on*, vol. 32, no. 1, pp. 102-115, 2014.

[13] Chinthagunta Mukundha, P.Gayathri and I.Surya Prabha “Load Balance Scheduling Algorithm for Serving of Requests in Cloud Networks Using Software Defined Networks”, Research India Publications, IJAER, Vol .11, No. 6 pp. 3910-3914 ,2016

[14] W. Sehery and T.C. Clancy, “Load balancing in data center networks with folded-Clos architectures,” *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, IEEE, 2015.

[15] M. Al-Fares, A. Loukissas and A. Vahdat, “A scalable, commodity data center network architecture,” ACM SIGCOMM

Comput. Commun.Rev., vol. 38, no. 4, pp. 63-74, 2008.

[16] A. Fischer et al, “Virtual network embedding: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, 2013.

[17] Amazon Virtual Private Cloud, [Online]. Available: https://aws.amazon.com/vpc.

[18] X. Meng, V. Pappas, and L. Zhang, “Improving the scalability of data center networks with traffic-aware virtual machine placement,” *IEEE INFOCOM*, 2010.