Multi-Tenant Network Acceleration Scheme for OpenStack

Linh (Paul) Phan, Kaikai Liu
Computer Engineering Department
San Jose State University (SJSU)
San Jose, CA, USA
Email: {Linh.phan, kaikai.liu}@sjsu.edu

Abstract—**Cloud virtualization and multi-tenant networking provide Infrastructure as a Service (IaaS) provider a new and innovative way to offer the on-demand services to their customers, such as the easy provisioning of new applications and the better resource efficiency and scalability. However, existing data-intensive applications require more powerful processor and computing power, as well as a high bandwidth, low latency and consistent networking service. In order to boost the performance of computing and networking services, as well as reduce the overhead of the software virtualization, we propose a new data center network design based on OpenStack, which is a promising cloud operating system solution. Specifically, we map the OpenStack networking services to the hardware switch, and perform hardware-accelerated L2 switch and L3 routing to solve the software limitations, as well as achieve the software-like scalability and flexibility. We designed our prototype system via the Arista Software-Defined-Networking (SDN) switch, and evaluated the performance improvement in terms of the bandwidth and delay using various tools. Our experimental results demonstrate that our datacenter networking solution achieves higher bandwidth, lower latency, and lower CPU utilization of the host server.**

Index Terms—**OpenStack, Arista, EOS, Neutron, LinuxBridge, Cloud Computing, Cloud Virtualization**

I. Introduction

The internet today is data driven with around 90 Exabyte of data flowing on the internet every month in the form of videos, images and other enriched content [1]. The demand for robust virtualized data center architecture is rising, small and large businesses are accepting cloud at a tremendous rate with around 175 billion dollars of revenue generated from public cloud services in the year 2016 [2].

Infrastructure as a Service (IaaS) is an instant computing infrastructure, provisioned and managed over the Internet. Quickly scale up and down with demand, and pay only for what users need, cloud computing is becoming a more efficient, low-cost, flexible and scalable service for companies and always in high demand. For networking, speed, performance, and availability are the essential requirements, hence it is critical to choose a robust cloud computing software that satisfies all the requirements, with minimal cost, fast deployment, easy management and scalability. Among variety choices on the market, OpenStack provides a solid set of software tools for building and managing cloud computing platforms for public and private cloud, with scalability and flexibility.

As the network grows and network traffic requires more processing power, faster speed and lower latency, software cannot handle much workload due to their limits and constraints. Therefore, it is essential to replace or enhance OpenStack vanilla software with alternative solutions, by taking advantages of its flexibility. In order to utilize the virtualized resources and improve the network performance, many researchers have proposed solutions to accelerate the network. Authors in [3], [4] propose to utilize the Mellanox FDR InfiniBand interconnect and Altera FPGA to accelerate the network and integrates with OpenStack. These approaches improve the speed at the cost of additional expensive hardware. In the mean time, using additional network devices also changes the existing cloud network topology and has the compatibility issues for normal applications.

In this paper, we present a practical solution to accelerate the networking performance in the cloud for data-intensive applications. Our solution does not need additional network hardware, e.g., InfiniBand or FPGA, and does not need to change existing cloud network infrastructure and software interface. Specifically, we utilize the hardware-based packet processing features in existing SDN-enabled switches, and integrate with the popular open-source Cloud management software, OpenStack Neutron. Through our framework and software drive, any network changes in the OpenStack dashboard will be automatically mapped to the physical SDN switch for hardware-based processing. Such integration will truly mix the benefits of software defined networking and hardware-accelerated processing together. It provides a scalable network architecture for multi-tenant data-intensive applications in the Cloud, and also facilitates the network programmability of OpenStack networking using APIs and dashboard.

II. Methodology

The benefits of using a physical L3 switch to replace the networking services that are handled by software in OpenStack, such as vRouter for layer 3, and Linux Bridge or OpenVSwitch for virtual switching, is to offer to the customers more power, speed, performance, reliability as well as scalability. For a small scale network, software switching and routing can handle the workload with acceptable efficiency, which are perfect for a lab or local cloud environment. However, as the network scales up and traffic is more heavy and more intensive, which are commonly seen in data center or cloud IaaS service provider, hardware networking devices will be more reliable, and are able to offer more power, faster speed and lower latency.

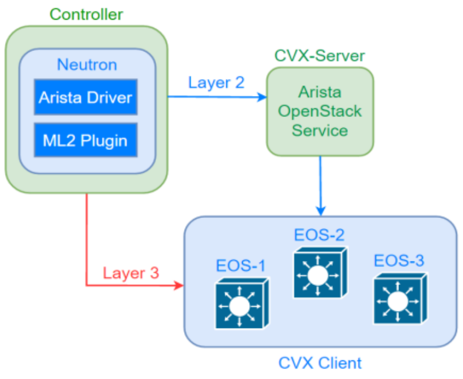


Fig. 1. Neutron ML2 Plugin - CVX Overview.

OpenStack Neutron networking service offers Modular Layer 2 (ML2) plugin, which is a framework that allows vendors to implement their own driver to work along with OpenStack layer 2-3 services. Arista EOS has extensive integration with the OpenStack Neutron project, giving customers a powerful network platform on which to run OpenStack deployments. By leveraging the Arista ML2 driver and Layer 3 service plugin, operators can automatically provision tenant networks across the physical infrastructure.

The Arista ML2 mechanism driver enables Neutron to automate VLAN provisioning on Arista switches, through Arista CloudVision Exchange (CVX). As shown in Fig. 1, CVX Server communicates with OpenStack Neutron on Controller node, and manages corresponding VLAN on the EOS switches as well as keep a network topology of available physical switches. Through LLDP, the switches are aware of what compute nodes are connected.

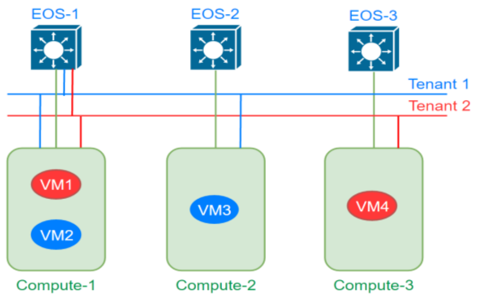


Fig. 2. Arista EOS - Compute Nodes Overview.

As VM instances are created on compute nodes, the Ethernet trunk port between the ToR and compute node is automatically configured to allow the required VLANs. As shown in Fig. 2, the Arista ML2 mechanism driver provisions VLANs in parallel with the virtual switch driver, that configures the VLANs on the virtual switch on the hypervisor host, and provides tight integration between network and compute provisioning.

The Arista L3 service plugin replaces the existing Neutron layer 3 service plugin. While Layer 2 need CVX to communicate with Neutron, L3 service plugin directly creates switched virtual interfaces (SVIs) on TOR switches when a virtual router is created in Neutron. Once configured, the hardware switch becomes the default gateway for the VMs, and all routing can be done in hardware on the switch, instead of in software at the Neutron network node.

III. Implementation

A Lab environment has been setup for testing and verification. The setup includes two core components:

Main Server: runs core OpenStack services, and works as the gateway for users to access via Horizon Web interface for instances management.

Arista L3 Switch: a 7050x EOS switch is used as a Top of Rack switch, also as a router for different network within OpenStack. In addition to the core components, there will be additional components that are used for testing, which are optional, including:

Rack Server: a server with high memory and strong processor, which can be used as an extra compute node for bigger test cases.

Provider host: a computer that lays on the provider network, for testing accessibility and managing instances.

B. Topology and Setup An OpenStack Newton environment has already built on top of the Server with KVM Hypervisor, The Servers and all Virtual Machines are running Ubuntu 16.04.2, including OpenStack core nodes as following:

Controller node: manage services such as Identity with Keystone, Images with Glance, Compute with Nova and Networking with Neutron. Controller node will also run the web based GUI Horizon for users and admin to connect and manage their instances as well as network.

Compute node: utilized for computing service, whom resources such as memory and processor will be shared between instances.

Controller Node and Compute Nodes are installed based on the documentations on OpenStack official website, with all the configuration and installation steps are written into an executable scripts for quick deployment under 1 hour.

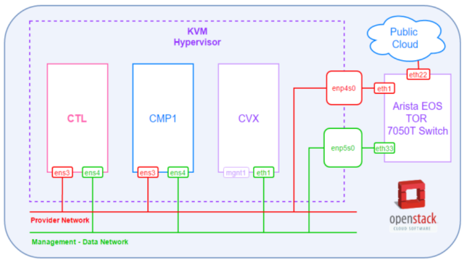


Fig. 3. Network Layout Overview.

Beside OpenStack core nodes, theres also a VM running vEOS as a virtual switch to work as CVX-Server. Fig. 3 shows the network layout of our experiment. All VMs is bridged to the physical host ethernet port, which are connected to a physical Arista EOS L3 switch, which will be used as CVX-Client for managing L2 and L3 service of OpenStack.

The CVX-Server will be used as the middleware for communicating between physical EOS switches and OpenStack Neutron via EOS API (eAPI). The CVX Server will get notified on new network creation from Neutron, and distribute the information to CVX-Client. The server also keeps a topology of available physical switches and hosts, as well as OpenStack instances.

IV. Experimental Evaluation

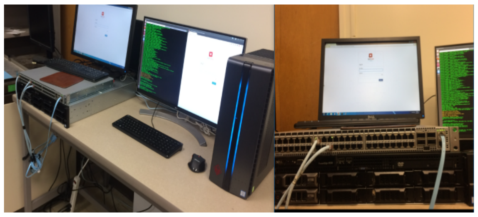


Fig. 4. The experimental environment.

The experimental environment is shown in Fig. 4. Neutron network controller node runs both OVS plugin and the Arista driver. Arista driver uses EOS API (eAPI) to communicate with Cloud Vision eXchange (CVX), which has the entire view of the Openstack cloud and communicates with the launched VMs in the compute node. The figure also has two provider networks, tenant 1 and tenant 2 with VLAN 1 and VLAN 2 automatically configured on the switch. OVS plugin conveys the VLAN choices to the Arista Driver via Cloud Vision Exchange. The Arista driver keeps CVX/Arista switch updated with the termination or relaunching on the VMs on other compute nodes or when the new provider networks are created which in turn updates VLANs created on the switch interfaces. In case of failure, CVX has the ability resync its state.

A. Test Scenario

In order to verify, a topology has been setup with test scenario: one tenant - two network - one router.

- When an internal network is created through either CLI or the web GUI Horizon, a new network is confirmed to be seen on CVX-Server and a Dynamic VLAN is created on EOS switch.
- When a virtual router is added, it is directly created on EOS switch as a SVI, and the IP is the default gateway of the private network it connects to.

B. Performance Test

Performance test will be performed in order to compare Layer 2 and Layer 3 differences when using OpenStack with vanilla networking service and with Arista EOS ML2 plugin.

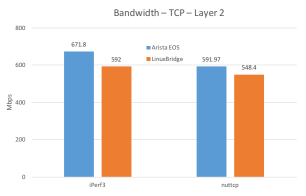
- Layer 3 testing will be performed between instances that lays in different network, and traffic has to go through vRouter.
- Layer 2 testing will be performed between two instances on the same network, but each on a different compute hypervisor to force traffic to go through the Switch as Layer 2 service

Each test will be performed 5 times, then calculate the average for result statistic.

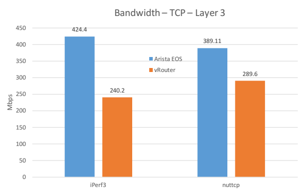
C. Result

Several performance tests were performed using various tools to generate bidirectional traffic from a host (client) lays in the internal network, to another server in the DMZ subnet. The test will focus on the differences in bandwidth and delay in both Layer 2 and Layer 3 when using software and hardware as networking services. The test will use three networking measurement tools:

- iPerf3: a tool that can generate TCP and UDP traffic for bandwidth measurement
- nuttcp: a tool that can be used for bandwidth and delay/RTT measurement with TCP/UDP traffic
- mytraceroute(mtr): an advance version of traceroute for delay/RTT measurement.



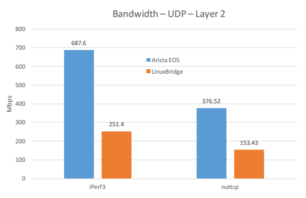
(a) L2



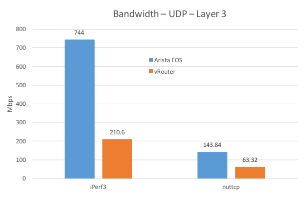
(b) L3

Fig. 5. Bandwidth result for TCP traffic of: (a) L2; (b) L3.

For TCP Protocol as shown in Fig. 5, EOS shows a performance boost about 100-150Mbps for both Layer 2 and Layer 3 traffic. For setup with EOS, the bandwidth can reach 671Mbps for layer 2 and 424.4Mbps for layer 3 in average. For setup with OpenStack vanilla vRouter and LinuxBridge, the bandwidth can reach 591.97Mbps for layer 2 and 289Mbps for layer 3. Hence, for TCP traffic, EOS setup definitely increases the bandwidth performance.



(a) L2



(b) L3

Fig. 6. Bandwidth result for UDP traffic of: (a) L2; (b) L3.

For UDP traffic of Fig. 6, the tools focus on flooding the packets out, hence resulting in high packet loss. For layer 2 and layer 3, EOS setup shows almost triple bandwidth performance compared to LinuxBridge/vRouter setup. This is because EOS setup has the hardware power to do packet routing, while vanilla setup has to process routing with software which has limited performance.

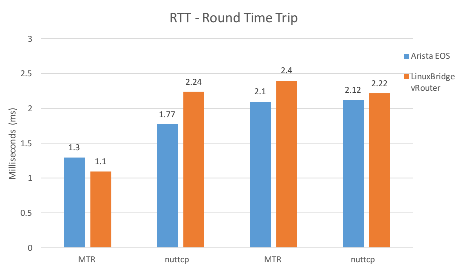


Fig. 7. RTT results for TCP traffic.

The round time trip (RTT) are also calculated using both tools mytraceroute and nuttcp as shown in Fig. 7. EOS and Linux Bridge shows similar result with less than 1 millisecond different for both layer 2 and layer 3.

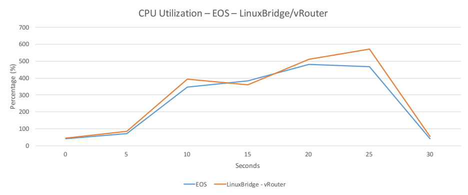


Fig. 8. CPU utilization - EOS vs. Vanilla Setup.

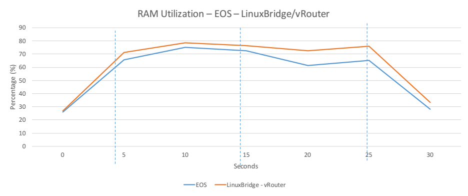


Fig. 9. Memory utilization - EOS vs. Vanilla Setup.

The CPU and RAM utilization statistics for EOS and Vanilla Setup with Linux Bridge/vRouter are shown in Fig. 8 and Fig. 9. Both are measured for hypervisor machines, with all NICs set to promiscuous mode to pass all traffic it receives to the CPU rather than passing only the frames that the controller is intended to receive. The statistics is monitored with hypervisors idle in second 0-5, and iperf3 script executed for layer 2 in second 5-15, follow by layer 3 in second 15-25, and idle for second 25-30.

For layer 2, because this is a small scale test with two VMs in separated compute hypervisors, the Linux Bridge shows better CPU utilization than EOS. This is because in EOS setup, when traffic has to travel between two instances that in two different compute hypervisor, the packets are forced to travel through EOS switch, hence the hypervisor need to processes the same amount of traffic twice, inbound and outbound. Meanwhile, in LinuxBridge setup, traffic only needs to be processed once within the hypervisor, and will cost less CPU once the cache is updated. They both use a similar amount of RAM, with EOS costs less RAM since packets does not need to be processed locally.

For layer 3, the CPU utilization for EOS setup is better than vanilla vRouter setup. In vRouter setup, when there is a router involve, this will need extra process power on the controller with networking service, while in EOS setup, all traffic will be passed to the switch and processed by the switch hardware. Therefore, EOS setup also consume less RAM than vRouter setup because the packets will be processed and stored in the switch memory rather than in the hypervisor memory.

The results and statistics above prove that Arista Switch setup has better performance with faster speed and lower latency than using Linux Bridge, as well as having less overhead on CPU and RAM utilization. This is very important when the network scales up, because one or few controller node will not be enough to process a huge amount of traffic. Hence, the results can be concluded that, using hardware devices has better efficiency and performance in routing and forwarding between instances, and using the design, datacenter can improve their cloud service for accelerating data intensive applications.

V. Conclusion

The Arista OpenStack solution provides a number of ways for administrators to orchestrate their Arista switches. The ML2 plugin automates the provisioning of VLANs on Arista switches, and with the Arista layer 3 service plugin, a hardware switch can serve as the routing gateway. The ability to orchestrate the physical network devices provisioning within the OpenStack solution is a significant achievement for the market.

Improving the efficiency and performance in datacenter and cloud service provider is always an important factor, which can be done by achieving software flexibility and hardware-based efficient in routing and forwarding. The design of combining OpenStack Neutron and hardware layer 3 switches can eliminate the limits and constraints of software, while ensuring performance, reliability, flexibility and scalability, which are required to accelerate data intensive applications.

Acknowledgment

This work is funded by Arista Network. The author is deeply grateful for the research opportunities, as well as the funding support and technical guidance from Arista throughout the project process.

References

[1] The zettabyte era trends and analysis. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html

[2] Facts and statistics about cloud computing. [Online]. Available: https://www.statista.com/topics/1695/cloud-computing/

[3] P. Rad, R. V. Boppana, P. Lama, G. Berman, and M. Jamshidi, "Low-latency software defined network for high performance clouds," in *System of Systems Engineering Conference (SoSE), 2015 10th*. IEEE, 2015, pp. 486–491.

[4] X. Ge, Y. Liu, D. H. Du, L. Zhang, H. Guan, J. Chen, Y. Zhao, and X. Hu, "Openanfv: Accelerating network function virtualization with a consolidated framework in openstack," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 353–354, 2015.