2018 IEEE 11th International Conference on Cloud Computing

OpenStack Network Acceleration Scheme for Datacenter Intelligent Applications

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) providers a new and innovative way to offer on-demand services to their customers, such as easy provisioning of new applications and better resource efficiency and scalability. However, existing data-intensive intelligent applications require more powerful processors, higher bandwidth and lower-latency networking service. In order to boost the performance of computing and networking services, as well as reduce the overhead of software virtualization, we propose a new data center network design based on OpenStack. Specifically, we map the OpenStack networking services to the hardware switch and utilize hardware-accelerated L2 switch and L3 routing to solve the software limitations, as well as achieve software-like scalability and flexibility. We design our prototype system via the Arista Software-Defined-Networking (SDN) switch and provide an automatic script which abstracts the service layer that decouples OpenStack from the physical network infrastructure, thereby providing vendor-independence. We have evaluated the performance improvement in terms of bandwidth, delay, and system resource utilization using various tools and under various Quality-of-Service (QoS) constraints. Our solution demonstrates improved cloud scaling and network efficiency via only one touch point to control all vendors' devices in the data center.**

***Keywords*-OpenStack, Arista, EOS, Neutron, LinuxBridge, Cloud Computing, Cloud Virtualization**

I. Introduction

The internet today is data driven, with around 90 exabytes of data flowing 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 the cloud at a tremendous rate, with around 175 billion dollars of revenue generated from public cloud services in the year 2016 [2].

The current data center design has transformed to a software defined network architecture with virtualization forming the crux of all the newly commissioned data centers. Speed, performance, and availability are the essential requirements of the network. It is critical to choose a robust networking framework that satisfies all the requirements, with minimal cost, fast deployment, easy management, and scalability. However, the networking performance of the existing cloud is not on the same caliber as that of traditional high-performance computing (HPC), which typically uses InfiniBand for the networking [3]. However, contemporary HPC grids and clusters cannot adapt to dynamic workloads or allocate resources to concurrent multi-tenant applications [4], [5], [6]. Software defined networks paradigm helps solve this problem by isolating layer 2/3 functionalities within tenant networks by using OpenvSwitch (OVS) or Linux bridges. Software switches form an integral part of any virtualized computing setup. They provide network access for Virtual machines [7]. The most popular virtual switch implementation, OpenvSwitch (OVS), is heavily used in cloud computing platforms like OpenStack. However, this solution does not match performance levels of traditional networking as obtained through dedicated network hardware. This is due to the multiple dependencies of packet processing in the software, which significantly hinders performance.

In this paper, we present a practical solution to accelerate networking performance in the cloud for data-intensive applications. Our solution does not require additional network hardware, e.g., InfiniBand or FPGA, nor does it rely on changing existing cloud network infrastructure or software interfaces. 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 driver, any network changes in the OpenStack dashboard are automatically mapped to the physical SDN switch for hardware-based processing. Such integration truly combines the benefits of software-defined networking and hardware-accelerated processing. This provides a scalable network architecture for multi-tenant data-intensive applications in the cloud and also facilitates network programmability of OpenStack networking using APIs and the dashboard.

II. System Design

Building a private, public or hybrid OpenStack cloud requires virtual and physical network infrastructure that is programmable, agile and resilient. We propose to deliver a highly scalable and automated cloud infrastructure for an OpenStack environment via an automatic script. By leveraging our proposed Enhanced Switch Offloading for OpenStack Network (ESOO), customers will be able to drastically speed up business services, mitigate operational complexity, and reduce costs. The architecture of our proposed "ESOO" is shown in Fig. 1. Our ESOO framework focuses

2159-6190/18/$31.00 ©2018 IEEE
DOI 10.1109/CLOUD.2018.00146

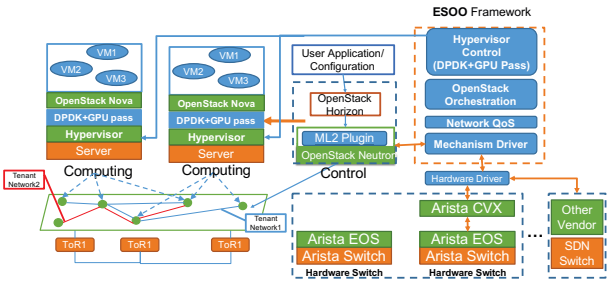


Figure 1. System Overview.

on improving the OpenStack networking performance by substituting the LB/OVS packet switching with hardware-based switching. At the same time, this new setting will not impact the flexibility of the network automation or pose any compatibility issues. Our framework provides a complete view of the network (topology) and its current and consistent state to the underlying hardware switch. When an OpenStack user performs any networking related operation (create/update/delete/read on network, subnet and port resources), the Neutron server receives the request, passes the same to the configured plugin and makes the appropriate change to the DB. Our ESOO also leverages the Intel's Data Plane Development Kit (DPDK) [8] to reduce the computational overhead when accessing the network interface card (NIC) through the Hypervisor. When the GPU is available in the server and users' application requires GPU acceleration, our ESOO framework will setup the GPU pass-through for this tenant automatically. By performing hardware related configuration and setup automatically, users can get the maximum performance out of the data center in a transparent way.

III. Offloading the Packet Processing to the Hardware Switch

The proposed ESOO framework substitutes the software-based LB/OVS packet switching to hardware-based switching and thereby improves the performance of the OpenStack network. With the integration of OpenStack Neutron and SDN switch, the user-triggered changes to the network and network elements in the OpenStack Dashboard (Horizon) are translated into Neutron APIs and handled by neutron plugins. Our ESOO bridges the neutron plugins and corresponding agents running in the hardware SDN switch. This thin layer is compatible with the existing functions in the OpenStack Neutron, and also supports external SDN controllers (such as Open Daylight and RYU) through the ML2 plugin.

A. OpenStack Orchestration on Switch Cluster

To interface the networking service (Neutron) of OpenStack to the hardware switch (using Arista's 7050T), we leverage the OpenStack Mechanism driver and communicates with Arista's CloudVision eXtension (CVX) over the Command API (eAPI) [9], [10]. CVX is able to view the entire switch cluster from Arista. Through eAPI, the CVX can appropriately provisions VLANs on switch interfaces so that the compute instances (OpenStack Nova) on the compute nodes have end-to-end connectivity over tenant networks.

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 OVS for virtual switching, offers customers more power, speed, performance, and reliability, as well as scalability. For a small-scale network, software switching and routing can handle the workload with acceptable efficiency, which is perfect for a lab or local cloud environment. However, as the network scales up and traffic becomes heavy and intensive, which is commonly seen in data centers or cloud IaaS service providers, hardware networking devices will be more reliable and are able to offer more power, faster speed, and lower latency.

B. Cloud and Network Topology

We propose a prototype data center design with the following four components: 1) **Main Server**: Runs core OpenStack services, and works as the Horizon dashboard for users to manage instances. The Main Server is also the controller node, which is the central hub for all the commands executed. The controller node hosts many OpenStack servers such as Keystone for authentication, Glance for image storage, neutron for networking, and a few others. This node lets the user interact with the different supported APIs and the dashboard. Many other components such as Database, Object Storage, and Block Storage can be separated into other autonomous nodes. 2) **Arista Switch**: Two 7050x EOS switches are used as a Top of Rack (TOR) switches, also as a router for different networks within OpenStack. 3) **Rack Servers**: Two Dell R630 servers with high memory and powerful processor are used as computing nodes for AI applications. 4) **CVX Server**: One server is running vEOS to work as the CVX-Server. The hardware switch works as the CVX-Client. The CVX-Server is 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. 5) **Provider host**: one computing node (computer) as the gateway is used to access the provider network for testing accessibility and managing instances. The prototype system environment is shown in Fig. 2.

A Neutron network controller node runs both the OVS plugin and the ML2-based driver for our proposed ESOO. Our ESOO can be interfaced with different vendors' hardware switches. The OpenStack ML2 interface makes the

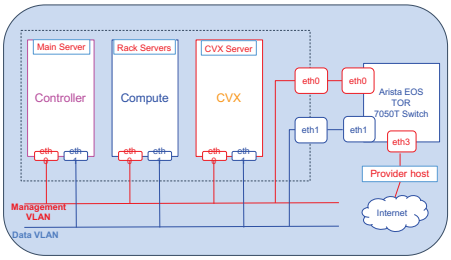


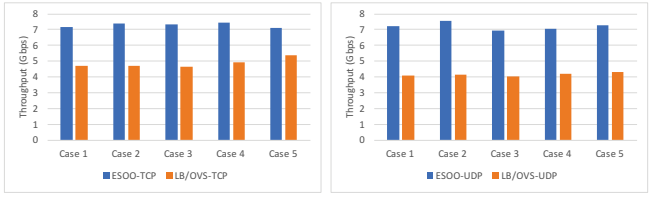
Figure 2. The system topology of our prototype private cloud.

vendor-independent design feasible. In this paper, we utilize Arista's SDN switch as one example. Our OpenStack driver uses Arista's EOS API (eAPI) to communicate with Arista's Cloud Vision eXchange (CVX), which has the entire view of all the connected Arista switches. In the case of failure, CVX has the ability to resync its state. The server also keeps a topology of available physical switches and hosts, as well as OpenStack instances. Using CVX simplifies the communication with the hardware switch. For example, only one interface is needed for Arista's switch or another interface to connect to other vendors' hardware without connecting each switch individually. To reduce the overhead when accessing the network interface card (NIC) and GPU in the VM, we apply the Intel DPDK and GPU pass-through in the hypervisor to setup the direct data path.

IV. Experimental Evaluation

A. Throughput Evaluation

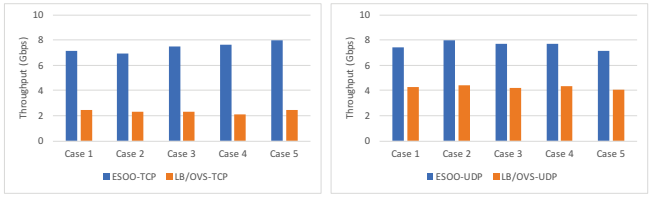
Fig. 3 shows the Layer-2 TCP and UDP throughput results based on iPerf3. From these two Layer-2 throughput results, we can see our proposed ESOO can reach from 7Gbps to almost 8Gbps for both TCP and UDP traffic. With LinuxBridge/OVS setting in existing OpenStack, the maximum throughput can only reach 4.96Gbps in Layer 2 switching. The throughput difference is due to our ESOO setup having enough hardware power to efficiently perform the packet switching, while the existing OpenStack network setup has to process switching using software with limited performance.



(a) TCP (b) UDP

Figure 3. The achieved L2 throughput of two different approaches based on iPerf for the (a) TCP traffic and (b) UDP traffic via iPerf.

Fig. 4 shows the Layer-3 TCP and UDP throughput results based on iPerf. OpenStack utilizes the vRouter for Layer-3 routing, which is very flexible for various software-defined network topologies. However, the achieved maximum TCP throughput is limited to 2-2.2 Gbps, and the achieved maximum UDP throughput is limited to 4-4.3 Gbps due to the software limit. In contrast, our hardware-accelerated solution (ESOO) achieves around 7-8 Gbps throughput for both the TCP and UDP traffic. Comparing the TCP and UDP traffic, the TCP traffic has the delivery guarantee, hence the results for TCP are more realistic comparing to the UDP traffic. This is also why the UDP throughput is higher than that of TCP; UDP traffic is only for testing the maximum amount of bandwidth that an interface can handle.



(a) TCP (b) UDP

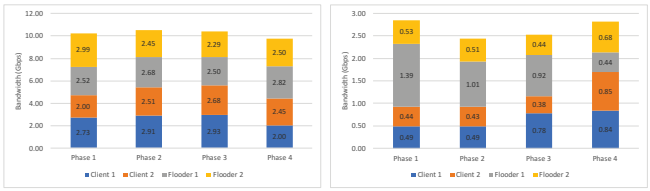
Figure 4. The achieved L3 throughput of two different approaches based on nuttcp for the (a) TCP traffic and (b) UDP traffic via iPerf.

B. Quality of Service (QoS) Test

To evaluate the Quality of Service (QoS) and estimate the impact of the QoS from other traffic, we have a more complex topology with multiple nodes and different traffic patterns. Our test topology includes 6 nodes as follows: 1) Client-1 and Client-2 are located in Net-1 with QoS policy applied, and they send traffic to Server-1 and Server-2, respectively; 2) The Server-1 is located outside the current datacenter and can be accessed via the provider-network and router "EOS-R2"; 3) The Server-2 is located in Net-2 and connected with the client via the same router "EOS-R1"; 4) "Flooder" is located on Net-4 with two parallel 5Gbps traffic to Server-1 and Hypervisor-1, respectively. We use "Flooder" to model other network traffics that consume the available bandwidth. To maintain the QoS policy, the OpenStack Neutron will have to dynamically allocate the network bandwidth for all networks/instances. We utilize this process to emulate a realistic datacenter network environment and evaluate the effectiveness of our ESOO scheme under the QoS scenario.

We apply one QoS policy on the Net-1 for both Client-1 and Client-2 in the same network. The QoS policy includes 2 rules: 1) **a maximum bandwidth of 4Gbps** and 2) **a minimum bandwidth of 2Gbps**. When the QoS policy is applied, the OpenStack Neutron ensures the network will have at least 2Gbps for each ports/instance it connects to, and will not exceed a maximum of 4Gbps. To evaluate the dynamic performance impact of the QoS policy on Client-1 and Client-2, we designed a four-phase benchmark traffic

pattern with 30 seconds for each phase. For Client-1, traffic phase 1 is 4 Gbps; phase 2 is 3 Gbps, phase 3 is 4 Gbps, and phase 4 is 2 Gbps, i.e., a "4-3-4-2 Gbps" pattern. For Client-1, the traffic phase 1 is 2 Gbps, phase 2 is 3 Gbps, phase 3 is 3 Gbps, and phase 4 is 4 Gbps, i.e., a "2-3-3-4 Gbps" pattern. The flooder has two parallel links flooding 5Gbps traffic out (10 Gbps in total). We performed the same test using both the ESOO setup and OpenStack LinuxBridge/vRouter setup, and compared the differences in performance.



(a) ESOO (b) LB/vRouter

Figure 5. The CPU usage results of 3Gbps traffic on different machines for: (a) our proposed 'ESOO'; (b) OpenStack's LinuxBridge/vRouter setting.

For a maximum 10Gbps link between physical servers and switches, the bandwidth is distributed among networks and instances. Using our proposed 'ESOO' scheme, all hosts under QoS policy are ensured the guaranteed 2Gbps minimum bandwidth and 4Gbps maximum bandwidth as shown in Fig. 5a. The OpenStack Neutron ensures the QoS policy and no instance/network can occupy all the bandwidth. The Neutron QoS scheme reduces the Flooders' traffic bandwidth to ensure the QoS policy of Client-1 and Client-2 under the maximum 10Gbps link constraint. The QoS results of the LinuxBridge/vRouter setup are shown in Fig. 5b. As the bandwidth results of Client-1 and Client-2, it can be seen that the bandwidth under the QoS policy cannot be guaranteed due to software limits. The available bandwidth pool that shared among all network/instances is less than 3Gbps. When the traffic in the network is higher than the limit, the Service Level Agreement (SLA) cannot be achieved and each instance within the QoS policy can only have a 0.5 Gbps bandwidth.

V. Conclusion

The OpenStack network solution provides a number of ways for administrators to orchestrate their cloud environment. In this paper, we present a practical solution to accelerate the networking performance of the OpenStack for data-intensive intelligent applications. Our proposed ESOO framework can help automatic provisioning multi-tenant networks and fully exploit the capabilities of the existing hardware (i.e., NIC, GPU, and SDN Switch) without requiring additional hardware accelerators. Our experimental results demonstrate that our solution can push the hardware to its limits (i.e., 10Gbps network throughput) under various topology. Our solution helps in orchestrating the private cloud via a highly integrated three-step script while ensuring performance, reliability, flexibility and scalability that are required to accelerate data-intensive applications.

Acknowledgment

The work presented in this paper is funded by Arista Networks, Inc. and National Science Foundation under Grant No. CNS 1637371.

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] N. S. Islam, M. Rahman, J. Jose, R. Rajachandrasekar, H. Wang, H. Subramoni, C. Murthy, and D. K. Panda, "High performance rdma-based design of hdfs over infiniband," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 35.

[4] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE'08*. Ieee, 2008, pp. 1–10.

[5] J. Jose, M. Li, X. Lu, K. C. Kandalla, M. D. Arnold, and D. K. Panda, "Sr-iov support for virtualization on infiniband clusters: Early experience," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*. IEEE, 2013, pp. 385–392.

[6] Q. He, S. Zhou, B. Kobler, D. Duffy, and T. McGlynn, "Case study for running hpc applications in public clouds," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 2010, pp. 395–401.

[7] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar *et al.*, "The design and implementation of open vswitch." in *NSDI*, 2015, pp. 117–130.

[8] Dpdk boosts packet processing, performance, and throughput. [Online]. Available: https://www.intel.com/content/www/us/en/communications/data-plane-development-kit.html

[9] Research brief: Arista cloudvision: managing switches as software. [Online]. Available: https://www.arista.com/assets/data/pdf/Analysts/

[10] Arista - cloudvision cvx overview. [Online]. Available: https://www.arista.com/en/cg-cv/cloudvision-cvx-overview