






Characterization of Network Management Traffic in OpenStack based on Virtual Machine State Changes

Adnei W. Donatti¹^a, Charles C. Miers¹^b, Guilherme P. Koslovski¹^c, Maurício A. Pillon¹^d
and Tereza C. M. B. Carvalho²^e

¹Graduate Program in Applied Computing (PPGCA), Santa Catarina State University (UDESC), Brazil

²Laboratory of Sustainability in ICT, University of São Paulo (USP), Brazil

Keywords: Characterization, Cloud Computing, Network Traffic, OpenStack.

Abstract: OpenStack is a popular and versatile solution for creating IaaS clouds. OpenStack has several private cloud issues implementations regarding its network infrastructure with which organizations are not always familiar. In this context, this work characterizes the administrative network traffic of OpenStack clouds. Administrative traffic has a separate network, which can affect the performance of the cloud as a whole. We set up an induced lifecycle for virtual machines (VMs) and measured network traffic and Application Programming Interface (API) calls for each task with different operating system (OS) images. Moreover, we also provide an analysis and characterization of the measured network traffic in the management security domain of OpenStack, as well as verify the possibility of using a linear regression model for predicting the traffic volume produced for each task.

1 INTRODUCTION


OpenStack (<https://www.openstack.org>) is one of the most popular private and public IaaS cloud solutions. OpenStack allows administrators to customize their network configuration and induces the networks of the cloud should be divided into three security domains: public, guest, and management (OpenStack, 2019b). This configuration aims to guarantee basic traffic isolation and security along with the cloud network. Moreover, traffic isolation is necessary for preventing cloud administrative operations from affecting the user's network performance and degrading the performance of hosted applications.


The network infrastructure in a data center is essential for satisfactory performance (Maswood and Medhi, 2017). If the network is slow, some cloud-hosted service may be slow too (OpenStack,). Cloud administrators need to plan this infrastructure correctly to avoid performance problems / bottlenecks. OpenStack allows the customization of the service


module distribution within the data center. The servers and all service module can be placed following the administrator's objective (e.g., high availability, consolidation, and load balancing). In this context, we discuss how common VM management tasks may impact the administrative network of an OpenStack-based cloud. In this sense, a network traffic analysis and characterization are performed. We share some of the fundamental concepts of (Donatti et al., 2020), but we expressively deepened the discussion on traffic characterization with 8 more operating systems, different flavors, and using a completely new monitoring tool developed by us.


The research gap here addressed is about the lack of information regarding how user-generated tasks (e.g., creating an instance of VM) may impact on the most internal network domain of OpenStack. The main contributions of the present paper are: (i) the characterization of administrative network traffic based on user's actions (VM states changes); (ii) experimental results considering multiple OS images; and (iii) linear regression to support the estimation of traffic useful for bandwidth management.


This work is organized as follows. Section 2 defines the network-related concepts of OpenStack clouds, and Section 3 discusses the related work. Section 4 presents the characterization method, while

^a <https://orcid.org/0000-0002-4085-9640>

^b <https://orcid.org/0000-0002-1976-0478>

^c <https://orcid.org/0000-0003-4936-1619>

^d <https://orcid.org/0000-0001-7634-6823>

^e <https://orcid.org/0000-0002-0821-0614>

Section 5 details the testbed, experimentation processes, and results. Section 6 discusses the analysis, and Section 7 considerations / future directions.

2 OpenStack INFRASTRUCTURE

OpenStack works as a cloud operating system for a large pool of computational resources (OpenStack, 2019d), controlling all data center servers. Inter-service communication is commonly performed through a messaging queue service, but REST requests can also be executed. The messaging queue services implement the Advanced Message Queuing Protocol (AMQP), usually a RabbitMQ solution (OpenStack, 2020). It is worthwhile to highlight, the messaging queue service is fundamental for OpenStack even though there is no specific module for these services.

The data center (DC) network design and configuration for OpenStack clouds may change according to the demand of the cloud administrator. Although there are several ways to configure a DC network for OpenStack, there are a few common points, which must be considered. OpenStack documentation states the division of the network traffic into security domains: public, guest, and management (Figure 1)(OpenStack, 2019a).

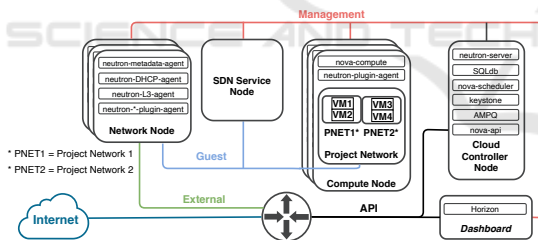


Figure 1: Standard OpenStack networking setup (Donatti, et al., 2020).

The Public Domain is formed by the API and External networks. The External network provides Internet access to VMs, while the API network is used to access OpenStack APIs. The Guest Domain network is basically the only network in the Guest Security Domain used by VM communication within the cloud deployment; and the Management Domain is the most internal security domain reachable only within the data center. The Management Domain is composed by the Management network, but may also includes the Database network. OpenStack components communication as well as the access to VM images and volumes, for example, are held over the Management Security Domain.

3 RELATED WORK

The cloud infrastructure analysis is often seen from the user’s perspective (Aishwarya. K and Sankar, 2015; Shete and Dongre, 2017; Chaudhary et al., 2018; Alenezi et al., 2019), relinquishing the internal operations and behavior of the cloud provider. There is a lack of information regarding how user generated tasks (e.g., VM launch) may impact the behavior of the management network (Donatti. et al., 2020). Besides, cloud performance can be evaluated by analyzing its behavior while its usage (Bruneo, 2014).

This paper offers the use of an analysis and characterization approach for the understanding of the network traffic into the provider’s management network regarding consumer submitted tasks in VMs (e.g., creating, stopping, shelving instances of VMs). This network traffic understanding helps cloud administrators to better design all the cloud architecture elements (e.g., network topology, bandwidth). In this sense, we defined five criteria which are used to compare this work to the other works in this area (Table 1).

Table 1: Related work comparison.

Criteria	(Sankari et al., 2015)	(Flitner and Bauer, 2017)	(Gustamas and Shidik, 2017)	(Venzano and Michiardi, 2013)	(Sciammarella et al., 2016)	(Donatti. et al., 2020)
Collect traffic on the OpenStack cloud management network	Partially. Focus on analyzing the SDN traffic of DCs	No	Yes	No	Yes	Yes
Classify the network traffic regarding the state changes of VMs	No	No	No	No	Partially. Only VM creation and termination	Yes
Analyze the collected traffic for identifying which service the packets are related	No	No	No	No	No	Partially. Significant "MISC" traffic and RabbitMQ traffic was still not characterized
Store the characterized traffic into a database	N.I.	N.I.	No	N.I.	N.I.	Yes
Identify the timing in which packet was collected (timestamp)	Yes	N.I.	No	N.I.	Yes	Yes

4 CHARACTERIZATION METHODOLOGY

Traffic analysis and characterization are both approaches, which can be used to understand and solve performance issues in computer networks (Dainotti et al., 2006), being generally composed by two steps: (i) measurement: the collection of data being transmitted over the network; and (ii) traffic analysis: realized to identify/classify relevant characteristics. The traffic measurement can employ tools to capture data traveling across the network (e.g., TCPdump). Depending on how measurement is performed, it can be classified as (Williamson, 2001): Active (network traffic creation by the monitoring system, inducing specific situations), and Passive (capture only existing network traffic).

Among the classification techniques (port-based, statistical, pattern matching, and protocol decoding) commonly used to classify internet traffic (Dainotti

et al., 2012; Finsterbusch et al., 2014), a port-based approach fits well when characterizing the OpenStack management network. All the services running in this context are supposed to use well defined ports, e.g., Nova API (compute services) uses port TCP/8774. However, when deploying a naive port-based approach, the traffic generated by inter-services communication is masked as RabbitMQ network traffic, since it uses RabbitMQ’s port (TCP/5672) and not the application port itself. For improving the network traffic characterization, we adopted a different strategy to characterize the packets. A reviewed port-based approach is used, we also run *lsof* GNU/Linux utility filtering the RabbitMQ’s port. This approach allows to identify which process is listening to RabbitMQ during the network traffic collection. Moreover, we adopted an active measurement of the consumer operations on a VM instance. Since we found no information to serve as a baseline for operations on VM instances, we chose the Active approach and defined the sequence of operations, called here as induced VM lifecycle. Each task/operation the user performs to a VM may cause change in its state, e.g., when the user shuts off a VM, the operation here is STOP and the VM will go to the STOPPED state as result.

The VM status cannot be measured at once, as an absolute representation of the VM. For understanding the status of a VM, OpenStack has three different types of states for VMs (Wiki, 2014). The combination of these states provides a more precise information about the VM current status: TASK_STATE provides information about ongoing tasks, e.g., when a user creates a VM, then the ongoing task is CREATING. VM.STATE means the current stable (not transition) state, e.g., if a VM is created then it achieves the ACTIVE state. POWER.STATE reflects a snapshot of the hypervisor state and reveals if the machine is still running and if there was a failure (e.g., RUNNING, SHUTDOWN). In this article, when the term VM state is used it refers to VM.STATE. OpenStack has a total of 12 possible VM states (OpenStack, 2018b). However, by analyzing the operations of users on our private OpenStack cloud, we find out the vast majority of our users typically have their VMs in only 6 states (Figure 2).

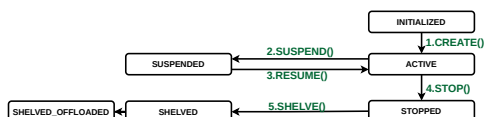


Figure 2: Induced VM lifecycle.

In Figure 2, when the VM is being created (operation CREATE is being performed), the VM will transition

from its initial state, INITIALIZED, to an ACTIVE state. Which means the VM is ready to be used. Once the VM is ACTIVE, we then suspend its operation, by performing SUSPEND procedure, which causes the VM to hit the SUSPENDED state. Next, the VM is reactivated by the RESUME operation, then it is stopped and shelved (STOP and SHELVE operations). The final state, SHELVED_OFFLOADED is hit when the OS image is no longer used by the hypervisor anymore. Therefore, this image may be discarded from the compute node. In our tests, we perform the induced life cycle 30 times each OS image. We collected traffic to identify the amount of data transmitted, the elapsed operation time, and the number of API calls for each state change enumerated in Figure 2. The collected data is stored in a database for further analysis.

5 TESTBED, EXPERIMENTS, AND RESULTS

CloudLab was chosen as our testbed for deploying OpenStack, Stein release. CloudLab provides researchers with 256GB RAM and two 2.4Ghz processors servers. The default OpenStack m1.small flavor, composed of 1 vCPU, 2GB RAM and 20GB storage, was adopted in all the experiments. All instances were interconnected by a 1Gb/s network link. A proof-of-concept cloud topology was prepared as described by Figure 3. The two nodes topology is enough to configure the networks (as stated in Section 2) and to identify the communication between nodes. In fact, we have several modules/services running on the controller node (here is also relevant network traffic into *loopback* interface), in which our monitoring tool runs, too.

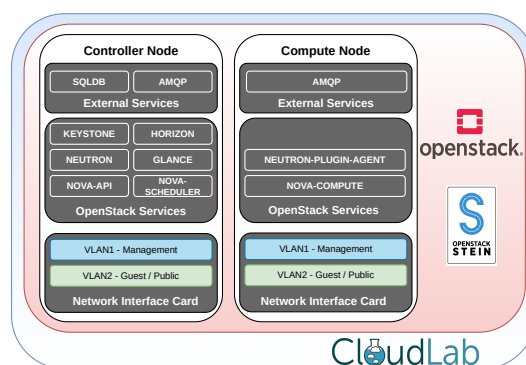


Figure 3: CloudLab testbed setup.

We created the OpenStack Network Monitor (ONM)⁶

⁶Developed in Python 3.6, using TCPdump for packet

tool organized into two main functions: (i) induced lifecycle - packet capture; and (ii) traffic analysis. ONM was customized to automate the experiments, being possible to specify VM parameters (e.g., image, flavor) and VM operations sequence (operations enumerated in Figure 2). It also supports working with VM image cache, which was not used in the experiments described in this article.

The induced lifecycle (Figure 2) is executed on 10 different OS images for instances of VMs. *FreeBSD* version 12.0, 454 MB image; *GNU/Linux Fedora Cloud* version 31-1.9, 319 MB image; *GNU/Linux Fedora Cloud* version 32-1.6, 289 MB image; *GNU/Linux Ubuntu Server* version 18.04 LTS (Bionic Beaver), 329 MB image; *MS-Windows Server* version 2012 R2, 6150 MB image; *GNU/Linux CirrOS* version 0.4.0, 15 MB image; *GNU/Linux CentOS* version 7, 898 MB image; *GNU/Linux CentOS* version 7, 1300 MB image; *GNU/Linux Debian* version 10, 550 MB image; and *GNU/Linux Ubuntu Server* version 20.04 LTS (Focal Fossa), 519 MB image. All images are QCOW2-based, except MS-Windows Server using QCOW2.GZ.

Table 2 shows a summary of data collected for all OSs. Each operation (CREATE, SUSPEND, RESUME, STOP, and SHELVE) was executed 30 times for each OS image of VM, and results are reported using the metrics: (i): Elapsed time of the operation execution; (ii): Total network traffic generated by the operation; and (iii): Total number of API calls identified on each operation.

All operations on the state of the VM have actions on the hypervisor, as well as on the Controller Node. However, an operation initiated at the Controller Node does not imply heavy network traffic. For example, a SUSPEND operation just removes the VM out of memory and releases the vCPUs, but the image file remains on the Compute Node. Thus, the results of some operations may vary depending on the system call implementation and we do not intend to evaluate those differences. The data collected shows that CREATE and SHELVE operations have the greatest impact on the volume of network traffic. This happens because there is a need to transfer the image from Glance service, inside the Controller Node (Figure 3), to the respective Compute Node at the time of CREATE. Likewise, in SHELVE the updated image in Compute Node needs to be transferred back to Glance service. Table 2 reveals the number total traffic to CREATE and SHELVE are not largely spread, between 0.02% to Windows Server and 2.17% to CirrOS, among the 30 observations. The rest of the VM's

collection, and employing OpenStack Python APIs: github.com/Adnei/openstack_monitor

Table 2: Data summary of the analyzed metrics.

Image	Operation	Total Traffic - MB (mean +/- sd)	Total API Calls (mean +/- sd)	Execution Time - seconds (mean +/- sd)
Ubuntu Bionic Beaver	CREATE	358.759 +/- 1.988	67 +/- 2.801	25.433 +/- 1.455
Ubuntu Bionic Beaver	RESUME	1.821 +/- 0.404	13 +/- 0.745	4 +/- 0
Ubuntu Bionic Beaver	SHELVE	1129.658 +/- 5.195	85 +/- 10.845	37.767 +/- 1.695
Ubuntu Bionic Beaver	STOP	2.291 +/- 0.874	16 +/- 2.985	9.8 +/- 0.761
Ubuntu Bionic Beaver	SUSPEND	2.359 +/- 0.773	10 +/- 1.717	6 +/- 0
Centos 7 (1300 MB)	CREATE	1419.955 +/- 0.709	100 +/- 2.541	48.6 +/- 1.776
Centos 7 (1300 MB)	RESUME	2.007 +/- 0.215	13 +/- 0.407	4.1 +/- 0.316
Centos 7 (1300 MB)	SHELVE	1424.304 +/- 0.639	108 +/- 1.442	57.2 +/- 1.229
Centos 7 (1300 MB)	STOP	1.518 +/- 0.21	6 +/- 0.651	4 +/- 0
Centos 7 (1300 MB)	SUSPEND	3.232 +/- 0.293	13 +/- 1.453	8.1 +/- 0.316
Centos 7 (898 MB)	CREATE	876.562 +/- 0.738	79 +/- 3.886	33 +/- 1.563
Centos 7 (898 MB)	RESUME	2.031 +/- 0.225	14 +/- 0.548	4 +/- 0
Centos 7 (898 MB)	SHELVE	936.403 +/- 0.453	82 +/- 3.806	39.8 +/- 1.317
Centos 7 (898 MB)	STOP	2.556 +/- 0.308	11 +/- 3.218	6 +/- 0
Centos 7 (898 MB)	SUSPEND	2.736 +/- 0.324	11 +/- 2.535	6.3 +/- 0.675
Cirros	CREATE	25.65 +/- 0.559	55 +/- 1.94	16.3 +/- 0.675
Cirros	RESUME	1.998 +/- 0.198	14 +/- 2.583	4 +/- 0
Cirros	SHELVE	33.749 +/- 0.271	31 +/- 2.833	7.2 +/- 0.632
Cirros	STOP	22.649 +/- 0.187	96 +/- 0.675	63.2 +/- 0.422
Cirros	SUSPEND	1.85 +/- 0.314	7 +/- 2.533	4 +/- 0
Debian 10	CREATE	592.787 +/- 0.676	78 +/- 2.644	32 +/- 0.667
Debian 10	RESUME	1.989 +/- 0.139	13 +/- 1.031	4 +/- 0
Debian 10	SHELVE	1539.051 +/- 0.808	119 +/- 2.062	63 +/- 1.563
Debian 10	STOP	2.339 +/- 0.509	10 +/- 3.059	5.6 +/- 0.843
Debian 10	SUSPEND	2.563 +/- 0.384	9 +/- 0.801	6 +/- 0
Fedora 31	CREATE	368.341 +/- 2.292	71 +/- 5.238	21.633 +/- 1.608
Fedora 31	RESUME	1.803 +/- 0.405	13 +/- 0.838	3.967 +/- 0.183
Fedora 31	SHELVE	993.256 +/- 3.7	73 +/- 9.138	30.033 +/- 0.669
Fedora 31	STOP	1.999 +/- 0.536	10 +/- 2.684	5.767 +/- 0.728
Fedora 31	SUSPEND	2.359 +/- 0.808	11 +/- 2.606	6 +/- 0
Fedora 32	CREATE	317.146 +/- 0.667	71 +/- 2.282	26.4 +/- 0.968
Fedora 32	RESUME	1.918 +/- 0.311	13 +/- 0.89	3.967 +/- 0.183
Fedora 32	SHELVE	853.332 +/- 0.797	83 +/- 4.452	40 +/- 1.145
Fedora 32	STOP	2.44 +/- 0.372	10 +/- 2.683	5.867 +/- 0.507
Fedora 32	SUSPEND	3.29 +/- 0.251	13 +/- 1.437	8.067 +/- 0.254
Ubuntu Focal Fossa	CREATE	555.52 +/- 1.143	75 +/- 4.109	28.8 +/- 1.229
Ubuntu Focal Fossa	RESUME	2.17 +/- 0.282	14 +/- 1.96	4 +/- 0
Ubuntu Focal Fossa	SHELVE	1391.565 +/- 0.695	118 +/- 3.347	61.7 +/- 1.947
Ubuntu Focal Fossa	STOP	4.318 +/- 0.607	17 +/- 3.161	10.2 +/- 0.632
Ubuntu Focal Fossa	SUSPEND	4.317 +/- 0.6	17 +/- 3.607	10.2 +/- 0.632
FreeBSD 12	CREATE	487.157 +/- 1.812	66 +/- 4.554	23.367 +/- 1.091
FreeBSD 12	RESUME	1.675 +/- 0.51	13 +/- 1.597	4.067 +/- 0.365
FreeBSD 12	SHELVE	483.855 +/- 1.243	44 +/- 3.809	15.533 +/- 0.507
FreeBSD 12	STOP	18.031 +/- 0.567	97 +/- 2.837	62.833 +/- 0.379
FreeBSD 12	SUSPEND	1.329 +/- 0.419	7 +/- 2.139	4 +/- 0
MS Windows Server	CREATE	6645.582 +/- 1.593	163 +/- 10.563	94.9 +/- 2.771
MS Windows Server	RESUME	1.829 +/- 0.368	13 +/- 1.234	4 +/- 0
MS Windows Server	SHELVE	6668.807 +/- 7.262	230 +/- 6.801	137.967 +/- 4.03
MS Windows Server	STOP	18.084 +/- 0.463	98 +/- 2.511	62.933 +/- 0.254
MS Windows Server	SUSPEND	1.24 +/- 0.433	6 +/- 1.517	4 +/- 0

state changes do not imply intensive network traffic, only API calls and local operations in Compute Node. For better presenting the OpenStack participation in the operation perform, we split the metrics network traffic volume and number of API calls by OpenStack service (Tables 3 and 4).

The number of observed API calls may vary according to the implementation of the induced lifecycle and VM configurations, e.g., additional configurations on the network would cause an increased number of Neutron-API calls. We used OpenStack Python APIs on our experiments, but there are other ways of performing those operations. OpenStack Connection.Compute (OpenStack, 2018a) was employed to run VM operations (e.g., CREATE, and SUSPEND). The results reveals the variation of API calls obtained was between 1.3% (CentOS) and 12.75% (Ubuntu Bionic Beaver). Although, CREATE operation of MS Windows Server has a Standard Deviation (SD) value a bit higher when comparing to the others SD values for all the operations and OS images. The highest SD value, in all operations, was caused by SUSPEND operation of CirrOS and MS Windows Server summarizing 36.2% and 34.9%, respectively. Table 4 presents API calls by services. Table 2 reveals CREATE and SHELVE operations are responsible for producing the highest amounts of traffic and those operations with longer time to finish.

Analyzing Table 3, Glance, the module which works as the manager of the VM images, is the responsible for most the network traffic. Also, each CREATE operation for all OS images produces

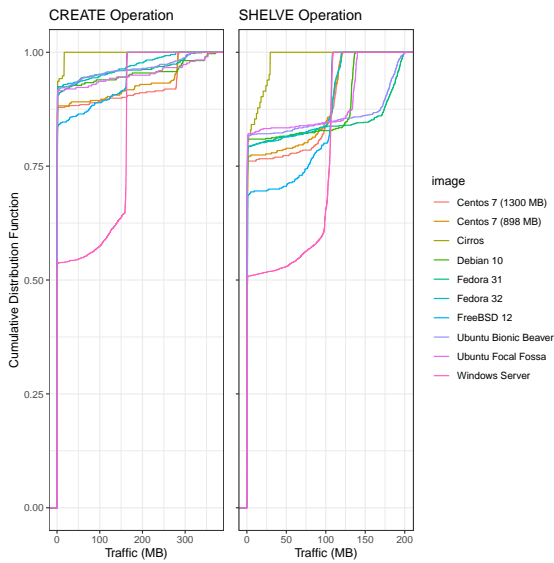


Figure 5: CDF plot of traffic per second for each OS image.

Figure 4 groups the network traffic volume (MB) per second (in $\log_{10} + 1$ scale) into box plots for each image of OS. The outliers identified for each OS image are due to peaks of network traffic per second when the image transfer is occurring. The same does not happen to MS Windows Server. The considerable size of MS Windows Server image implies a longer network transfer. In fact, it causes the network to keep high transfer rate long enough, so this high numbers are no longer listed as outliers of the distribution. This high numbers become common values inside this distribution, and thus, instead of outliers, it become part of the Q3 of the boxplot.

Figure 5 shows a Cumulative Distribution Function (CDF) for network traffic per second. CDF graphs allow to identify the peaks of data transfer through the network.

Figure 5 shows Fedora 31 and Ubuntu Bionic Beaver, for example, have most part of their traffic (around 90% in CREATE and 80% in SHELVE operation) transmitted at the beginning of the operation. This happens when the image is traveling through the network. The remaining operations, the network traffic gradually increases (with smaller peaks) until the operation is done. MS Windows Server operations analysis shows the beginning of both operations (CREATE and SHELVE), around to 50% of the total traffic was already transmitted. Sequentially, the CDF grows in an exponential way, which means the high tax of transmission (traffic per second) is kept until the operation traffic amount hits 100% and the operation ends. In order to study the relationship between the size of the OS image and the network traffic created by the operations to VMs we also set up a

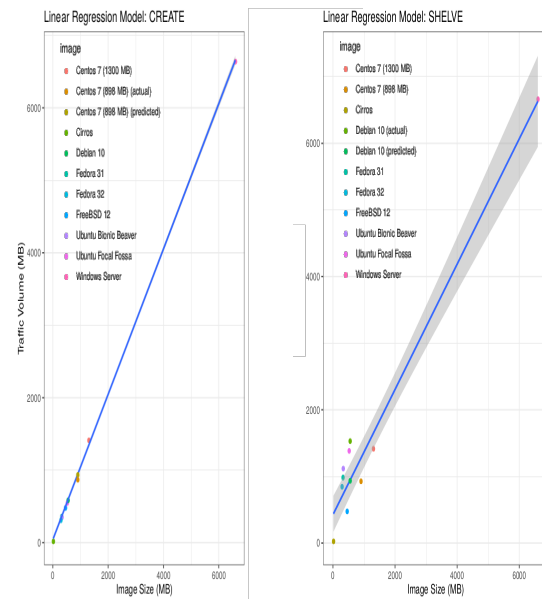


Figure 6: Linear regression model for CREATE and SHELVE operations. Image size is the predictor and network traffic is the target/response variable.

linear regression model. The linear regression model allows to understand the growth of the network traffic as a function of the image size. Therefore, we set the image size as a predictor for the network traffic, which is the target/response variable. Figure 6 shows the linear regression models for operations CREATE and SHELVE, $y = 40.700864 + 1.002707x$ and $y = 425.4478 + 0.9401x$, respectively. Y stands for the response variable (network traffic volume) and X stands for the predictor (image size in MB). We employed 9 OS images and 1 other image to compare the predicted value by the model to an actual measured value. The OS image used in the comparison predicted vs. actual is chosen randomly.

We found good accuracy responses for CREATE operation (Figure 6), such as a Min Max Accuracy (MMX) = 93% (approximately) and Mean Absolute Percentage Error (MAPE) = 7% (approximately). We adopted a confidence level of 90%; the identified values of intercept and slope: 40.700864 and 1.002707. Slope coefficients suggests there is a strong relationship between image size and network traffic (Pr value of $4e - 14$). Pr shows the probability of observing extreme values leading to coefficients of value 0 (called null hypothesis). If Pr is low enough, we can discard the null hypothesis. Thus, when the value of Pr is significant, it can be stated the null hypothesis is discarded. Regarding intercept coefficients, the relationship between image size and network traffic is not so strong despite still valid (Pr value of 0.0139); strongly significant R-squared and p-value: 0.9998

and $3.997e - 14$; and residual standard error of 32.3 MB on 7 degrees of freedom.

Regarding the linear model for SHELVE operation (Figure 6), we do not achieve high levels of accuracy: MMX = 61%; and MAPE = 39%. A 90% confidence level is adopted; we identified intercept and slope values of 425.4478 and 0.9401. Slope coefficients suggests there is a strong relationship between image size and network traffic (Pr value of $1.43e - 06$). Intercept coefficients suggests a valid relationship between image size and network traffic (Pr value of 0.0208). R-squared and p-value of 0.9697 and $1.425e - 06$, both significant for the context; and residual standard error of 366.5 MB on only 7 degrees of freedom. Overall, both linear models provide a direction of what to expect from the network traffic volume when performing CREATE and SHELVE operations. Also, it is evident that a larger dataset could lead the models to better results, in terms of statistically validation, since we already have good results.

6 ANALYSIS

CREATE and SHELVE operations produce most significant amount of network traffic. SUSPEND, RESUME, and STOP operations produce low volume of network traffic, which is very constant among our 30 observations (Table 2). Also, the networking traffic for these operations is highly related to the execution duration, since they basically rely on system calls and hypervisor actions, which are performed inside the Compute Node itself, as discussed in Section 5.

Most part of collected traffic was classified to the service it belongs to. The traffic created between inter-service communication handled by RabbitMQ was also listed to its respective services. There is still a small amount of MISC traffic (Table 3), which is related to MySQL, since several OpenStack modules were contemplated in the classification. Also, the number of API calls depends on how the operations are performed. There will not be a constant number of API calls for each operation. In fact, this may vary depending on configurations, such as the number of network interfaces on the VM instance.

Multiple instance creation or shelving are operations demanding a considerable amount of bandwidth. In a minimal topology, with Controller and Compute nodes only, for example, multiple instance creation and/or shelving at same time will cause the network to clog up. A basic topology, which has a dedicated storage network, deals better with multiple CREATE and SHELVE operations, since it has a separated physical network for storage node. Even though, depending on

the available bandwidth, this storage network will still be slow. Therefore, the prediction of network traffic volume generated by operations such as CREATE and SHELVE helps on solving the problem of bandwidth allocation to a storage network, for example.

OpenStack has a feature for caching images in Compute Nodes. Thus, when a new instance of the same type is launched on the same Compute Node, there is no need for transfer (network traffic) between Controller Node and Compute Node. In this sense, OpenStack adopts a standard policy for allocating a new instance of the "Worst Fit" type. For example, if a project with 10 instances of the same type is launched on the same Computer Node, then only one image will be transferred. However, this allocation policy can be changed to improve reliability and resilience. Reliability and resilience can be obtained by allocating each instance in a different Computer Node so that failure in one of them does not cause the service to drop all, but this will cause an increase in network traffic in the allocation (CREATE) and deallocation (SHELVE). Although image caching helps on reducing heavy network traffic, it does not fix totally the problem. Considering real scenarios, there may be several Compute Nodes distributed along the cloud data center, so the image must be transferred to them at least once. Also, this same image may be updated, what makes the image cache deprecated and new images may be created by the user. Thus, a baseline network traffic for operations such as CREATE and SHELVE may help on evaluating more complex scenarios, which demand several operations to be executed at same time.

Another aspect analyzed was the variation in the size of the instances' flavors. Initially we considered that an instance with more RAM could generate more traffic in the SHELVE operation. However, according to the OpenStack documentation (OpenStack, 2019c), in the SHELVE operation a STOP type operation is performed, implying the disposal of all information in the RAM memory. The only operation saving the information from RAM is a snapshot, which is not characterized by a change of a VMstate operation.

7 CONSIDERATIONS & FUTURE WORK

One of the key contributions of this work is to provide a baseline for the network traffic generated by the user's main operations of manipulating VMs in the OpenStack management network. We accomplished the characterization of administrative network traffic based on VM states changes. We also in-

cluded RabbitMQ network traffic to the characterization, which represents the inter-services communication inside the cloud architecture. Besides the linear regression model provides parameters to estimate the network traffic generated for any images in the QCOW2 format.

As future work, the latest versions of the MS Windows Server are being incorporated into the results. In addition, some variations in operations not included in the induced life cycle (presented in this work) are being measured.

ACKNOWLEDGMENTS

The authors would like to thank the support of FAPESC, LabP2D / UDESC, and CloudLab.

REFERENCES

- Aishwarya, K and Sankar, S. (2015). Traffic analysis using hadoop cloud. In *(ICIECS)*, pages 1–6.
- Alenezi, M., Almustafa, K., and Meerja, K. A. (2019). Cloud based sdn and nfv architectures for iot infrastructure. *Egyptian Informatics Journal*, 20(1):1 – 10.
- Bruneo, D. (2014). A stochastic model to investigate data center performance and qos in iaas cloud computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 25(3):560–569.
- Chaudhary, R., Aujla, G. S., Kumar, N., and Rodrigues, J. J. P. C. (2018). Optimized big data management across multi-cloud data centers: Software-defined-network-based analysis. *IEEE Communications Magazine*, 56(2):118–126.
- Dainotti, A., Pescape, A., and Claffy, K. C. (2012). Issues and future directions in traffic classification. *IEEE Network*, 26(1):35–40.
- Dainotti, A., Pescape, A., and Ventre, G. (2006). A packet-level characterization of network traffic. In *2006 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*, pages 38–45.
- Donatti, A. W., Koslovski, G. P., Pillon, M. A., and Miers, C. C. (2020). Network traffic characterization in the control network of openstack based on virtual machines state changes. In *Proc. 10th CLOSER*, pages 347–354. SciTePress.
- Finsterbusch, M., Richter, C., Rocha, E., Muller, J., and Hanssgen, K. (2014). A survey of payload-based traffic classification approaches. *IEEE Communications Surveys Tutorials*, 16(2):1135–1156.
- Flittner, M. and Bauer, R. (2017). Trex: Tenant-driven network traffic extraction for sdn-based cloud environments. In *2017 Fourth International Conference on Software Defined Systems (SDS)*, pages 48–53.
- Gustamas, R. G. and Shidik, G. F. (2017). Analysis of network infrastructure performance on cloud computing. In *2017 International Seminar on Application for Technology of Information and Communication (iSemantic)*, pages 169–174.
- Maswood, M. M. S. and Medhi, D. (2017). Optimal connectivity to cloud data centers. In *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, pages 1–6.
- OpenStack. What to do when things are running slowly. <https://docs.openstack.org/operations-guide/ops-maintenance-slow.html>. Accessed: 2020-08-13.
- OpenStack (2018a). Openstack docs: Connection. <https://docs.openstack.org/openstacksdk/latest/user/connection.html>. Accessed: 2020-07-15.
- OpenStack (2018b). Provision an instance.
- OpenStack (2019a). Networking architecture.
- OpenStack (2019b). Openstack documentation. <https://docs.openstack.org>. Accessed: 2020-07-15.
- OpenStack (2019c). Shelve and unshelve an instance.
- OpenStack (2019d). What is openstack? <https://www.openstack.org/software>.
- OpenStack (2020). Message queuing. <https://docs.openstack.org/security-guide/messaging.html>. Accessed: 2020-07-22.
- Sankari, S., Varalakshmi, P., and Divya, B. (2015). Network traffic analysis of cloud data centre. In *2015 International Conference on Computing and Communications Technologies (ICCCCT)*, pages 408–413.
- Sciammarella, T., Couto, R. S., Rubinstein, M. G., Campista, M. E. M., and Costa, L. H. M. K. (2016). Analysis of control traffic in a geo-distributed collaborative cloud. In *2016 5th IEEE Cloudnet*, pages 224–229.
- Shete, S. and Dongre, N. (2017). Analysis and auditing of network traffic in cloud environment. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 97–100.
- Venzano, D. and Michiardi, P. (2013). A measurement study of data-intensive network traffic patterns in a private cloud. In *Proc. IEEE/ACM 6th UCC, UCC '13*, pages 476–481, Washington/DC, USA. IEEE.
- Wiki, O. (2014). Vmstate. <https://wiki.openstack.org/wiki/VMState>. Accessed: 2020-07-22.
- Williamson, C. (2001). Internet traffic measurement. *IEEE Internet Computing*, 5(6):70–74.