SN Computer Science (2023) 4:450
https://doi.org/10.1007/s42979-023-01836-8

ORIGINAL RESEARCH

Towards to Characterization of Network Management Traffic in OpenStack-Based Clouds

Adnei W. Donatti[1] · Charles C. Miers[2] · Guilherme P. Koslovski[2] · Maurício A. Pillon[2] · Tereza C. M. B. Carvalho[1]

Received: 30 September 2021 / Accepted: 12 April 2023

Abstract
OpenStack is versatile and popular, allowing full customization for creating private or public IaaS clouds. This work addresses a network traffic analysis and characterization for the management domain inside OpenStack clouds. We conduct an induced lifecycle to execute virtual machine (VM)-related tasks, measure the traffic, and identify the services behind the traffic generated by these operations. Also, we analyze the impact of different images of operating systems (OSs) and VM's flavor in the measured traffic. Moreover, we observe that predicting the volume of network traffic from operations, such as creation and shelving instances of VMs, helps estimate bandwidth boundaries, avoiding bottlenecks, for example.

Keywords OpenStack · Network traffic · Characterization · VM · Virtual machine

Introduction

The network infrastructure is a key element for the cloud's performance [13]. When the network is slow, some cloud-hosted services may be affected too [14]. OpenStack clouds allow administrators to customize the network configuration. A typical configuration divides the network into three security domains: public, guest, and management [18]. This configuration aims to guarantee basic traffic isolation and security along with the cloud network, which is necessary for preventing cloud administrative operations from causing a negative impact on the user's network performance. For instance, the process of creating or shelving VMs instances may cause heavy administrative network traffic, which needs to be separated from the user domain [9].

Cloud administrators need to plan the cloud infrastructure correctly to avoid performance problems/bottlenecks. To cope with the infrastructure planning, OpenStack enables the customization of the distribution of the services within the data center. The servers and all service modules can be placed following the administrator's objective (e.g., high availability, consolidation, and load balancing). In this context, this paper carries a network traffic analysis and characterization, which gives insights on how common VM management tasks (e.g., creating, pausing, and shelving) may affect the administrative network of an OpenStack-based cloud. This work stands as an extended version of [7]. Such a modality of paper brings space for better discussing fundamental concepts, the experimentation process, and approaching new experiments. Therefore, we highlight the contributions of this work to understanding how one could carry a network traffic characterization in an OpenStack deployment, which is a crucial step for resource planning in the cloud. In this extended version, we include a dataset [8] summarizing the results of our experiments, and we also investigate the impact of the VM's flavor on the network traffic and present.

This paper aims at 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.

This work was funding by the National Council for Scientific and Technological Development (CNPq).

This article is part of the topical collection "Cloud Computing and Services Science" guest edited by Donald Ferguson, Markus Helfert, and Claus Pahl.

✉ Charles C. Miers
charles.miers@udesc.br

Adnei W. Donatti
adnei.donatti@usp.br

1 Graduate Program in Electrical Engineering, Escola Politécnica da Universidade de São Paulo, São Paulo 05508-010, Brazil

2 Graduate Program in Applied Computing, Santa Catarina State University, Joinville 89219-710, Brazil

Published online: 14 June 2023

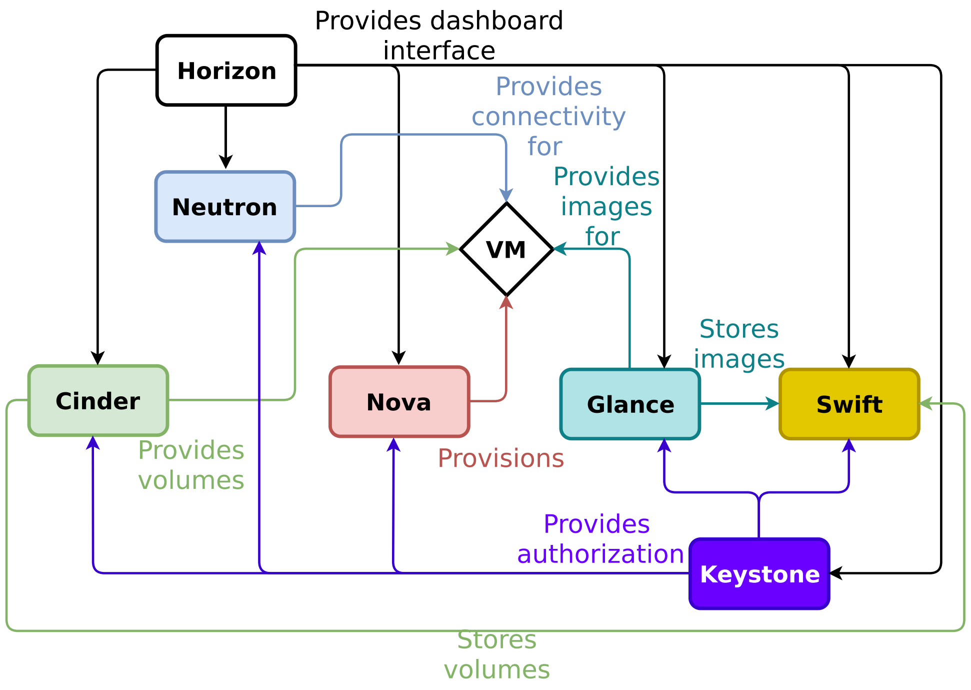


Fig. 1 Services interaction related do VM operation

The main contributions of this work are: (i) the characterization of management network traffic based on VM-task-related (e.g., creating and shelving); (ii) experimental results considering multiple OS images and flavors; and (iii) linear regression to estimate network traffic (useful to cover not experimented scenarios and for bandwidth management).

This work is organized as follows. "OpenStack infrastructure" defines the network-related concepts of OpenStack clouds, and "Related work" discusses the related work. "Characterization methodology" presents the characterization method, while "Experiments and results" details the testbed, experimentation processes, and results. "Analysis" discusses the analysis, and "Considerations and future work" presents our considerations.

OpenStack Infrastructure

OpenStack controls a large pool of computation resources, acting as an operating system for the cloud [19]. To do so, OpenStack divides the management services into optional or core modules. Core modules represent the essential ones for operating the cloud. For example, the networking functionalities are held into Neutron module as well as Nova module holds computing services. Also, these modules interact with each other atop the data center network [21]. Inter-service communication is commonly performed through a messaging queue service, but REST requests can also be executed.

The message queuing services are essential for the cloud to operate in a distributed manner, providing efficient inter-process communication [20]. OpenStack supports RabbitMQ, Qpid, and ZeroMQ solutions. ZeroMQ (https://zeromq.org/) works with direct peer-to-peer communication through TCP sockets, while RabbitMQ (https://www.rabbitmq.com/) and Qpid (https://qpid.apache.org/) implement the Advanced Message Queuing Protocol (AMQP). Traditional OpenStack deployments use RabbitMQ. Figure 1 exemplifies how different services may interact/communicate to execute VM-related tasks.

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 (Fig. 2) [17]. Moreover, some of the core OpenStack modules are:

- **Horizon** (dashboard): used for cloud overview and management.
- **Nova** (compute): handles mostly instance-related-tasks, e.g., initialization, scheduling, and deallocation of VMs;
- **Neutron** (network): provides network connectivity all over the cloud;
- **Glance** (image manager/storage): manages the storage and retrieval images of VMs and containers;
- **Swift** (object storage): responsible for the storage and retrieval of unstructured objects;
- **Cinder** (block storage): provides persistent block storage for running instances; and
- **Keystone** (identity): responsible for authentication and authorization services.

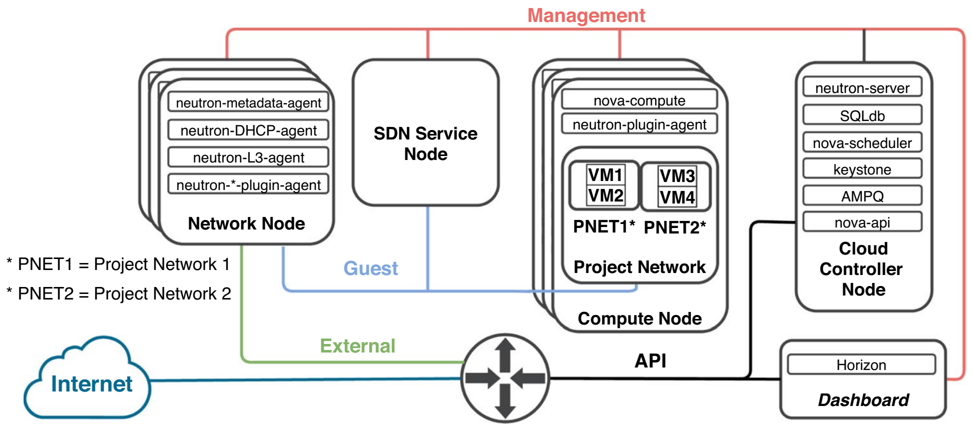


Fig. 2 Standard OpenStack networking setup [9]

Table 1 Related work comparison

Criteria	[23]	[11]	[12]	[26]	[24]	[9]
Collect traffic on the OpenStack cloud management network	Partially	No	Yes	No	Yes	Yes
Classify the network traffic regarding the state changes of VMs	No	No	No	No	Partially	Yes
Analyze the collected traffic for identifying which service the packets are related	No	No	No	No	No	Partially
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

The Public Domain is comprised of the Application Programming Interface (API) and External networks. The External network provides Internet access to VMs, while the API network is used to access OpenStack APIs. The Guest network is the one inside 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 mainly composed of the Management network, although it could also include a Storage network. OpenStack components' communication as well as the access to VM images and volumes, for example, are held over the Management Security Domain.

OpenStack provides users with several VM configuration options. The VM flavor describes the basic set of specifications about the VM. For example, one can define the storage volume for the OS, the RAM configuration, and the number of virtual CPUs. The VM flavor configuration must be carefully thought, considering the OS and the necessary resources for the machine to properly run. In turn, flavors can be tailored to CPU-intensive or RAM-intensive applications. The side effects of an imprecise flavor configuration may not be exclusive of the VM itself, as the data center network can also be impacted, since storage services, which will be holding snapshots, for example, are decoupled from the compute nodes. For example, if the user requests more disk space than the VM actually needs, disk-related operations will waste computing and networking resources.

Related Work

The cloud infrastructure analysis is often seen from the user's perspective [1, 2, 4, 25], 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 [9]. Besides, cloud performance can be evaluated by analyzing its behavior while its usage [3].

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 VM-related tasks performed by the user (e.g., creating, stopping, and shelving instances of VMs). This network traffic understanding helps cloud administrators to better design all the cloud architecture elements (e.g., network topology and bandwidth). In this sense, we defined five criteria which are used to compare this work to the other works in this area (Table 1).

In work [9], RabbitMQ traffic remained not characterized and there was a significant amount of miscellaneous (MISC) network traffic. In our previous work [7], it was shown the use of linear regression to predict the total network traffic volume produced by some user tasks for the management of VMs, as well as the significant amount of MISC (miscellaneous) traffic was reduced. In this paper, we focus on an extended version from [7], better detailing the experimentation process, presenting a new experiment with VM flavors, and deepening fundamental concepts. Finally, among the related work, [24] is similar to our proposal. However, the authors focused only on the network traffic generated by creating and destroying multiple VM instances in geo-distributed collaborative clouds, without separating traffic between services, nor do they try to identify the time to perform operations and the number of calls for each OpenStack service.

Characterization Methodology

Traffic and analysis characterization are techniques employed to understand and solve performance issues in computer networks [6]. Generally, these techniques involve two steps: (i) **measurement**: collection/measurement of data flowing through the network; and (ii) **traffic analysis**: to study the measured data. Analyzing the network traffic is an important step to identify/classify relevant characteristics, although it can be limited according to the employed measurement phase. Moreover, measuring traffic may assume employing tools to capture data traveling across the network (e.g., TCPdump). Depending on how measurement is performed, it can be classified as **Active**, as the monitoring approach impacts on the system being monitored or induces specific situations, or **Passive**, in which the monitoring does not influence the system [28].

Among the classification techniques (port-based, statistical, pattern matching, and protocol decoding) commonly used to classify internet traffic [5, 10], a port-based approach fits well when characterizing the OpenStack management network. Inside the context of the OpenStack management network, the services running are supposed to use well-defined ports (e.g., Nova API—compute services—uses port TCP/8774). The knowledge of these well-defined ports is also important when defining firewall rules. However, when deploying a naive port-based approach, the traffic generated by inter-services communication is masked as RabbitMQ network traffic (as we conclude from work [9]), since it uses RabbitMQ's port (TCP/5672) and not the application port itself.

To properly address inter-service communication, we focused on mapping established connections to RabbitMQ. Once we know which services are communicating over RabbitMQ and what TCP port are they using, the port-based approach is still valid. Thus, we upgrade the first employed port-based approach by running *lsof* (GNU/Linux list of files) and mapping connections to RabbitMQ, helping to identify the processes 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.

The induced lifecycle is composed of VM-related tasks that cause the instance to pass through a set of state changes. For example, when the user shutoffs a VM, the operation/task here is STOP, and the resulting state of the VM is STOPPED. The operations/tasks in the induced lifecycle are: (1) CREATE; (2) SUSPEND; (3) RESUME; (4) STOP; and (5) SHELVE. Therefore, the VM instance is (1) created, and then, its activity is (2) suspended, (3) resumed, 4 stopped (shutoff), and (5) shelved. The induced lifecycle starts with the VM instance creation and ends when the VM is shelved (meaning that it is stored for further use). Figure 4 depicts the induced lifecycle as well as the set of state changes involved in the process.

Tracking the state of a VM in real time is a complicated task, since it requires the knowledge of three different information: (i) ongoing tasks; (ii) current situation/status; and (iii) power (e.g., ON or OFF, RUNNING or SHUTDOWN). OpenStack maps ongoing tasks (i) as the TASK_STATE, indicating what is happening to the VM (e.g., SUSPENDING, RESUMING, and DELETING). Also, the TASK_STATE indicates a state transition, named based on the action being executed [27]. About the current situation/status (ii), OpenStack maps as VM_STATE, indicating a stable non-transition state (e.g., PAUSED, STOPPED, and SHELVED) [27]. On the other hand, the power (iii) is mapped as POWER_STATE, reflecting a snapshot of the hypervisor state, revealing if the machine is still running and if there was a failure (e.g., RUNNING, SHUTDOWN, and FAILED).

The VM states depicted in Fig. 3 refer to the VM_STATE, representing the stable state. OpenStack has a total of 12 possible VM states [16]. 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, comprised by the induced lifecycle (Fig. 4). Moreover, we often use the term VM state to refer to the VM_STATE (stable non-transition state).

Summarizing Fig. 4:

- **1:** Operation CREATE initializes the instance of VM (the instance goes from state INITIALIZED to ACTIVE);
- **2:** Operation SUSPEND suspends the instance's activity (once the operation is done, the VM state goes from ACTIVE to SUSPENDED);

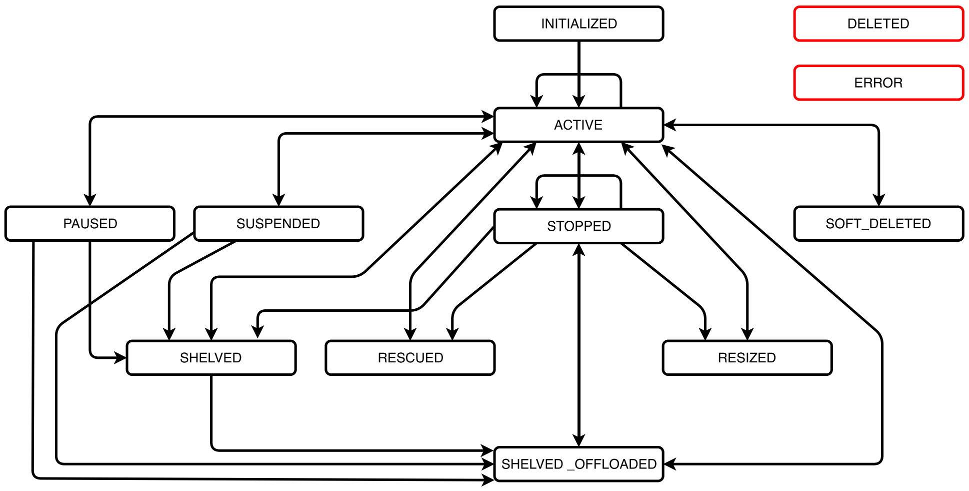


Fig. 3 OpenStack VM's states and transitions. DELETED and ERROR states are allowed to be reached from any other states [22]

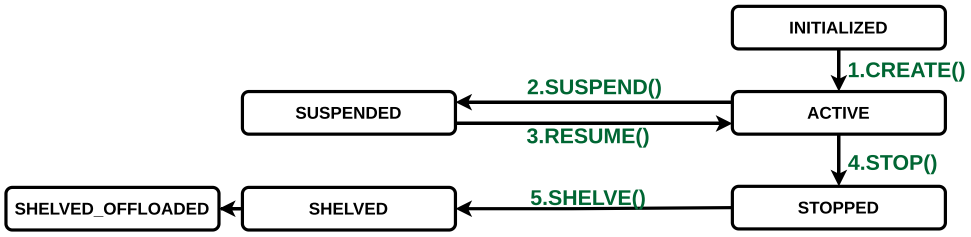


Fig. 4 Induced VM lifecycle

- **3:** Operation RESUME starts VM's activity from where it stopped (the VM state goes from SUSPENDED back to ACTIVE);
- **4:** Operation STOP performs a shutoff (the VM state goes from ACTIVE to STOPPED); and
- **5:** Operation SHELVE stores the instance for further use (the VM state goes from STOPPED to SHELVED and, once the hypervisor releases the VM's image, the final state hit is SHELVED_OFFLOADED).

Experiments and Results

In this section, we describe our experimentation methodology and results. Essential data are available through the dataset published on Zenodo [8].

Experiment Setup

CloudLab (https://www.cloudlab.us/) offers a flexible and isolated environment for research on cloud computing, and thus, it was chosen as our testbed for deploying OpenStack, Stein release. CloudLab provides researchers with 256GB RAM, and two 2.4 GHz processors servers. The OpenStack m1.small flavor, composed of 1 vCPU, 2 GB RAM and 20 GB storage, was set as a default flavor. All instances were interconnected by a 1 Gb/s network link. Figure 5 displays the deployment setup adopted over a two-node topology used in the experiments. The two-node topology is enough to configure and separate (using VLAN) all the network domains. Since this topology places several modules/services running on the controller node, the *loopback* interface is also relevant for monitoring.

The experiments are divided into: (i) OS **image changing**; and (ii) VM **flavor changing**. Experiment (i), OS image changing, tells us how the VM-related tasks behave (in terms of administrative traffic generated) according to the running OS in the machine. On the other hand, Experiment (ii), VM flavor changing, tells us whether the flavor choice itself arouses any network traffic impact. Both experiments use QCOW2-based OS images alongside KVM as the hypervisor (the default option in the CloudLab environment). However, adopting any other hypervisor (e.g., Xen), image file format, or OS image version does not impact the method or experiments. Nevertheless, on behalf of experiment replication, one should comprehend that using different image file formats may slightly change some procedures in the VM provisioning. Also, the experiments rely upon VM-related tasks described in the induced lifecycle (CREATE, SUSPEND, RESUME, STOP, and SHELVE), discussed in "Characterization methodology" and depicted in Fig. 4. Basically, Experiment (i) consists in performing the induced lifecycle against VMs running 10 different OSs, and Experiment (ii) consists in performing the induced lifecycle

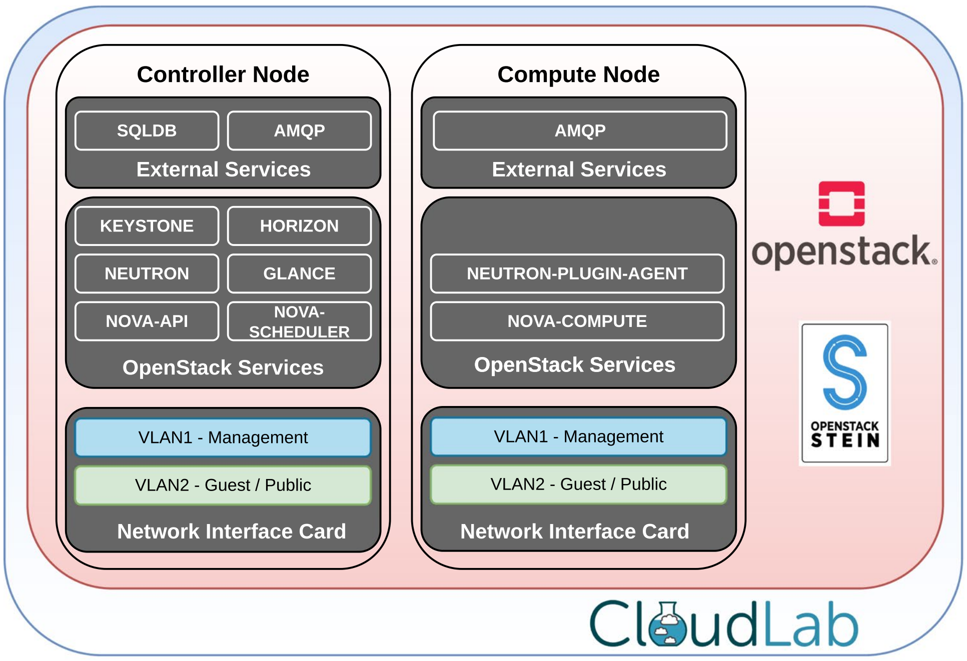


Fig. 5 Deployment setup adopted over two nodes on CloudLab testbed environment

against VMs running the same OS but with different flavor configurations.

Experiment (i), image changing, uses ten different QCOW2-based 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.

Experiment (ii), flavor changing, runs Ubuntu Bionic Beaver VMs created in four different flavors:

- *m1.small*: 1 vCPU, 20 GB Disk, and 2048 MB RAM;
- *m1.medium*: 2 vCPUs, 40 GB Disk, and 4096 MB RAM;
- *m1.large*: 4 vCPUs, 80 GB Disk, and 8192 MB RAM; and
- *m1.xlarge*: 8 vCPUs, 160 GB Disk, and 16384 MB RAM.

Automation Tools and Experiment Flow

To automate the experiments, we developed the OpenStack Network Monitor (ONM),[1] a tool that helps on measuring and analyzing the network traffic of OpenStack. ONM is divided into two main functions: (i) monitoring; and (ii) traffic analysis. It is possible to customize its operation, being able to fully parameterize the VMs (e.g., image and flavor), and specify a full set of VM-related tasks to be executed against the machines (we set the operations from the induced lifecycle), while the traffic monitoring runs (TCPdump-based). Moreover, ONM performs an analysis in the captured traffic, resulting in a database with relevant info about the traffic (e.g., the service/module and operation generating the traffic, size, and flow). Our tool also supports working with VM image cache, although it was not used in the experiments here described. The scheme in Fig. 6 depicts the experimentation flow.

ONM implements a module[2] that focuses on characterizing the network traffic from RabbitMQ ("Characterization methodology" introduces the challenges when characterizing RabbitMQ traffic). Since a naive port-based approach does not fit here, ONM also monitors RabbitMQ's port (TCP 5672) through *lsof*. In this way, all the established

[1] Developed in Python 3.6, combining TCPdump for packet collection, and OpenStack Python APIs for handling the VMs: `github.com/Adnei/openstack_monitor`.

[2] Service Identified: `github.com/Adnei/service_identifier`.

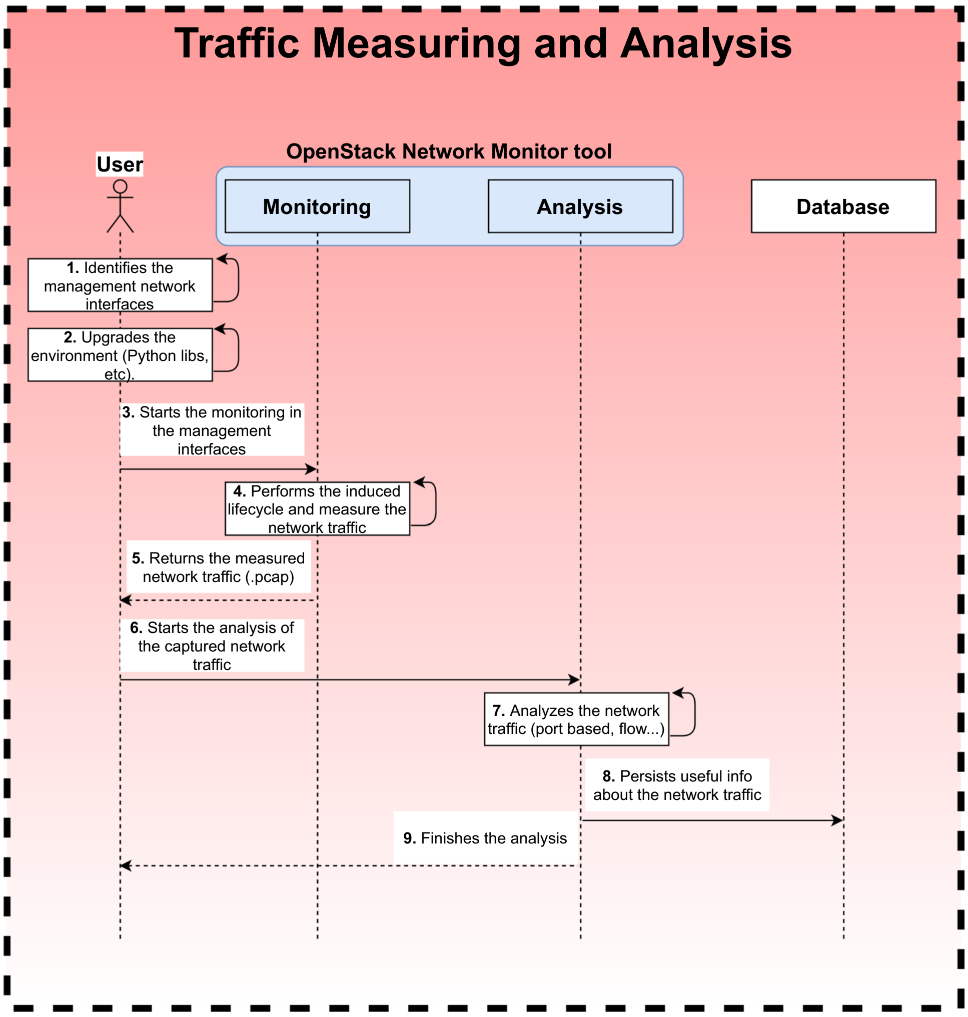
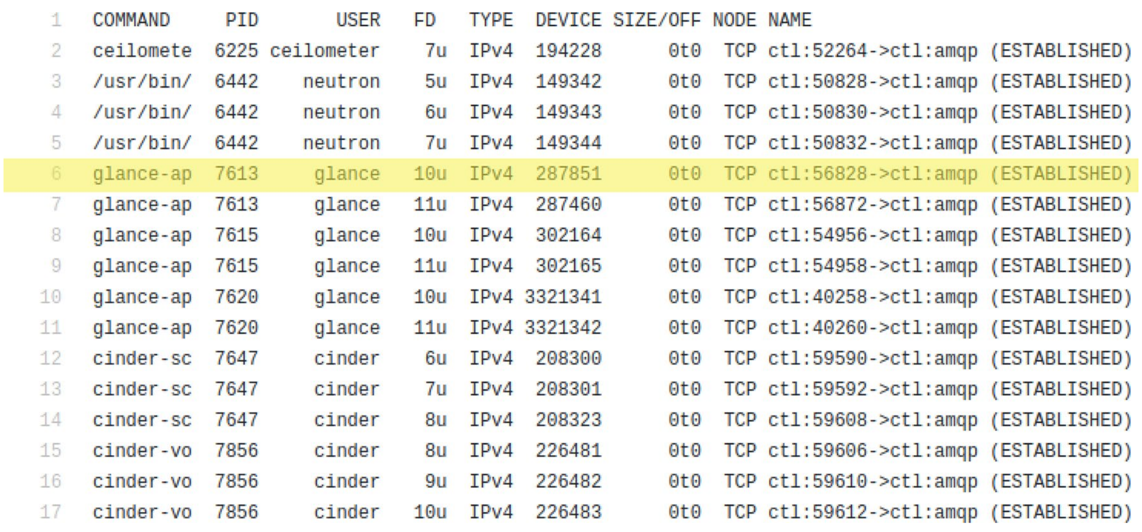

Traffic Measuring and Analysis
OpenStack Network Monitor tool
User
Monitoring
Analysis
Database
1. Identifies the management network interfaces
2. Upgrades the environment (Python libs, etc).
3. Starts the monitoring in the management interfaces
4. Performs the induced lifecycle and measure the network traffic
5. Returns the measured network traffic (.pcap)
6. Starts the analysis of the captured network traffic
7. Analyzes the network traffic (port based, flow...)
8. Persists useful info about the network traffic
9. Finishes the analysis

Fig. 6 Experimentation flow. The user informs the management network interfaces to ONM and the tool performs the induced lifecycle for each OS image. At the end of the process, a database is created holding all the useful info about the network traffic. The database is used for further study of the data (e.g., creating tables and plots)



```
COMMAND    PID       USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
ceilomete  6225 ceilometer    7u  IPv4  194228      0t0  TCP ctl:52264->ctl:amqp (ESTABLISHED)
/usr/bin/  6442    neutron    5u  IPv4  149342      0t0  TCP ctl:50828->ctl:amqp (ESTABLISHED)
/usr/bin/  6442    neutron    6u  IPv4  149343      0t0  TCP ctl:50830->ctl:amqp (ESTABLISHED)
/usr/bin/  6442    neutron    7u  IPv4  149344      0t0  TCP ctl:50832->ctl:amqp (ESTABLISHED)
glance-ap  7613     glance   10u  IPv4  287851      0t0  TCP ctl:56828->ctl:amqp (ESTABLISHED)
glance-ap  7613     glance   11u  IPv4  287460      0t0  TCP ctl:56872->ctl:amqp (ESTABLISHED)
glance-ap  7615     glance   10u  IPv4  302164      0t0  TCP ctl:54956->ctl:amqp (ESTABLISHED)
glance-ap  7615     glance   11u  IPv4  302165      0t0  TCP ctl:54958->ctl:amqp (ESTABLISHED)
glance-ap  7620     glance   10u  IPv4 3321341      0t0  TCP ctl:40258->ctl:amqp (ESTABLISHED)
glance-ap  7620     glance   11u  IPv4 3321342      0t0  TCP ctl:40260->ctl:amqp (ESTABLISHED)
cinder-sc  7647     cinder    6u  IPv4  208300      0t0  TCP ctl:59590->ctl:amqp (ESTABLISHED)
cinder-sc  7647     cinder    7u  IPv4  208301      0t0  TCP ctl:59592->ctl:amqp (ESTABLISHED)
cinder-sc  7647     cinder    8u  IPv4  208323      0t0  TCP ctl:59608->ctl:amqp (ESTABLISHED)
cinder-vo  7856     cinder    8u  IPv4  226481      0t0  TCP ctl:59606->ctl:amqp (ESTABLISHED)
cinder-vo  7856     cinder    9u  IPv4  226482      0t0  TCP ctl:59610->ctl:amqp (ESTABLISHED)
cinder-vo  7856     cinder   10u  IPv4  226483      0t0  TCP ctl:59612->ctl:amqp (ESTABLISHED)
```

Fig. 7 *lsof*—list open files—Linux command used to map connections established to TCP 5672 (RabbitMQ port) connections to the RabbitMQ port can be properly mapped, resulting in an efficient port-based approach. Figure 7 shows how a default "*lsof* -i:5672" output looks like. The third column tells us which service established a connection to the TCP 5672, and the last column tells us the source TCP port.

Summarizing the experimentation flow from Fig. 6, we use scripts to identify the management network interfaces, configure the environment, and configure ONM to run against these network interfaces and selected images. In the sequence, ONM performs the induced lifecycle and measures network traffic. The measured traffic is returned in the ".pcap" extension, which is used as input for the traffic analysis. Once the analysis is done, ONM returns a database with the relevant information only, such as packet source/destination, service, and timestamp.

Results

Table 2 brings a summary of data collected for all OSs in Experiment (i). Observed metrics are: (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. Each operation (CREATE, SUSPEND, RESUME, STOP, and SHELVE) was executed 30 times for each OS image of VM.

It is worthwhile to mention that VM-related tasks are basically handled at the Compute node, by the hypervisor, and the VM itself. 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. Also, a STOP operation, for example, depends on the OS running on the VM to be performed. Each OS may implement a shutoff system call in their own way, resulting in the most varying scenario. Thus, it is common that this kind of operation does not imply heavy network traffic, since there is no significant OpenStack participation in handling the operation itself, other than delegating it to the responsible parts (Compute node/hypervisor). However, operations, such as CREATE and SHELVE, rely on the active participation of OpenStack modules (which will be responsible for holding the VM image or snapshot), resulting in traffic being captured for analysis. In other words, the experiments are intended to measure the participation of OpenStack (in terms of administrative traffic generated) in the VM-related tasks.

In Table 2, CREATE and SHELVE operations have the greatest impact on the volume of network traffic. This happens, because the image needs to be transferred from Glance, in the Controller Node, to the Compute Node (Fig. 5). Likewise, in SHELVE, the snapshot taken in Compute Node needs to be transferred back to Glance. Table 2 also shows that the total traffic of CREATE and SHELVE is not largely spread among the 30 observations (between 0.02% to Windows Server and 2.17% to CirrOS). The rest of the VM-related tasks (SUSPEND, RESUME, and STOP) do not imply intensive network traffic, only API calls and local operations in Compute Node. To focus on the OpenStack participation, we split up the metrics by OpenStack service. Table 3 allows us to see the traffic by service, and Table 4 approaches the API calls by service.

Observing Table 4, the number of measured 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 automated the experiments using OpenStack Python APIs to handle VM-related tasks, as well as OpenStack Connection. Compute [15], but one could find another way to do so. 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 an 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, among all operations, was measured for CirrOS and MS Windows Server in SUSPEND operation, 36.2% and 34.9%, respectively.

From Table 3, it is evident that Glance, the module responsible for managing the VM images, represents most of the network traffic, as well as CREATE operation (for all images) produces the amount of Glance traffic around to the image size. For instance, the Glance traffic measured for CREATE operation using MS Windows Server is around 6615.876 MB, and the image size of MS Windows Server is 6150 MB, confirming the transmission of the image through the network. Therefore, one can assume the amount of administrative traffic as the total measured minus the image size. Proceeding with MS Windows Server CREATE example, 6645.582 MB (Table 2) − 6150 MB (image size) = 495.582 MB of administrative network traffic. SHELVE operation also takes the same logic, although the file transferred through the network is a snapshot, not an OS image. The remaining operations do not produce massive network traffic and run on few seconds.

Figures 8 and 9 provide a complimentary evaluation of the network behavior during CREATE and SHELVE operations. Figure 8 shows boxplots for the data flow per second (MB/s in log10 +1 scale). The boxplot assists in visualizing how spread the data are by dividing its "body" into four quartiles (dots represent outliers). For instance, the boxplots for Cirros and Centos 7 in CREATE operation look symmetric (equal proportions around the median), which means that it is a normal distribution. However, some identified outliers suggest peaks in the network traffic, possibly indicating the time when the image is transferred from one node to another through the network.

On the other hand, one may observe a positively skewed distribution by analyzing the boxplots for MS Windows CREATE and SHELVE (Fig. 8). In this case, the mean is

Table 2 Data summary of the analyzed metrics

Image	Operation	Total traffic, MB (mean ± sd)	Total API calls (mean ± sd)	Execution time, s (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	3.291 ± 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 ± 0.255	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	368.741 ± 2.292	71 ± 5.238	24.633 ± 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 ± 1.991
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	6668.887 ± 7.262	230 ± 6.801	137.967 ± 4.03
MS Windows Server	STOP	18.084 ± 0.463	98 ± 2.511	62.933 ± 0.254

Table 2 (continued)

Image	Operation	Total traffic, MB (mean ± sd)	Total API calls (mean ± sd)	Execution time, s (mean ± sd)
MS Windows Server	SUSPEND	1.24 ± 0.433	6 ± 1.517	4 ± 0

higher than the median. Also, there is such a frequency of high values (high data flow in the network) to the point of eliminating the outliers. Thus, the network is at a high rate during most of the operation, indicating the process of sending around 6 GB of data corresponding to the OS image. Additionally, the cumulative distribution provided in Fig. 9 reinforces the growing network traffic during about 45–50% of the operation.

Figure 9 shows that, for most OSs (apart from MS Windows Server and FreeBSD 12), the network behavior is constant for around 80% of the CREATE operation and 75% of the SHELVE operation. What happens to FreeBSD 12 is similar to the previously analyzed scenario on MS Windows Server. FreeBSD 12 has an execution time for SHELVE of around 15 s (Table 2). The only OS with a lower execution time for this operation is Cirros (about 7 s to a 15 MB image). SHELVE operation takes more than 30 s for all the other OSs, and MS Windows takes about 137 s, for instance (Table 2). In addition, Fig. 8 shows that FreeBSD also registers a high frequency of high values (positively skewed distribution in SHELVE operation). Nevertheless, it occurs within a shorter period. Figure 9 confirms the analysis, showing constant values during 68% of the operation (meaning a high data rate during 32% of the execution time for SHELVE).

We set up a linear regression model to study the relationship between the size of the image and the total traffic created by the operation. The linear regression model allows to understand the growth of the network traffic as a function of the image size. Therefore, the image size represents the predictor variable for the network traffic, which is the target/response variable. Figure 10 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 nine OS images and one 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 (Fig. 10), 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 that 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 (Fig. 10), 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 suggest that 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. Moreover, even with satisfactory results for the context, it is evident that a larger dataset could lead the models to a better statistical validation.

Another scenario worth investigating is the network traffic generated when the OS image remains the same, but the VM's flavor is changed, which is our Experiment (ii). As mentioned in "OpenStack infrastructure", the flavor of the VM specifies a basic set of configurations for the machine. Therefore, this scenario helps in understanding if the flavor choice may impact the network traffic volume. Table 5 summarizes the results for Experiment (ii), comparing metrics **Total Traffic** (in MB), **API Calls**, and 10 different QCOW2-based OS images for instances of VMs: for instances of VMs created under the flavors *m1.small*, *m1.medium*, *m1.large*, and *m1.xlarge*.

From Table 5, it is evident that the traffic volume does not change according to the flavor itself. However, one could measure the traffic volume by applying some load for memory and/or disk to the VM. Applying some load to the VM would actually make use of the resource allocated by the flavor and, perhaps, one could see the traffic volume increasing according to the flavor mostly for operation SHELVE. However, this is outside of the scope of this experiment.

Table 3 Traffic volume (MB)/service (mean ± SD)

Operation	Image	Ceilom-eter	Cinder	Designate	Glance	Heat	Keystone	Magnum	Manila	MISC	Neutron	Nova	Rab-bitMQ	Sahara	Trove
CREATE	Ubuntu Bionic Beaver	0.022 ± 0.004	0.008 ± 0.007	0.105 ± 0.012	346.61 ± 0.01	0.073 ± 0.087	0.14 ± 0.016	0.012 ± 0.019	0.007 ± 0.007	9.552 ± 1.605	0.345 ± 0.159	1.11 ± 0.194	0.249 ± NA	0 ± 0.001	0.011 ± 0.017
	Centos 7 (1300 MB)	0.022 ± 0.001	0.018 ± 0.015	0.195 ± 0.016	1398.295 ± 0.093	0.19 ± 0.186	0.195 ± 0.027	0.166 ± 0.345	0.006 ± 0.005	18.041 ± 0.919	0.902 ± 0.36	1.739 ± 0.449	0.257 ± NA	0.002 ± 0.002	0.158 ± 0.351
	Centos 7 (898 MB)		0.01 ± 0.006	0.142 ± 0.008	860.562 ± 0.123	0.223 ± 0.298	0.163 ± 0.027	0.245 ± 0.474	0.006 ± 0.007	13.229 ± 1.015	0.55 ± 0.293	1.362 ± 0.232	NA	0.001 ± 0.001	0.049 ± 0.055
	Cirros		0.003 ± 0.004	0.061 ± 0.004	15.784 ± 0.004	0.084 ± 0.142	0.145 ± 0.026	0.033 ± 0.049	0.001 ± 0.002	7.938 ± 0.404	0.434 ± 0.276	1.129 ± 0.04		0.001 ± 0.002	0.015 ± 0.031
	Debian 10		0.01 ± 0.007	0.137 ± 0.018	575.644 ± 0.005	0.362 ± 0.095	0.146 ± 0.023	0.033 ± 0.047	0.004 ± 0.004	14.482 ± 0.537	0.481 ± 0.164	1.366 ± 0.357	0.305 ± NA	0.002 ± 0.002	0.067 ± 0.061
	Fedora 31	0.021 ± 0	0.007 ± 0.006	0.116 ± 0.048	356.159 ± 0.017	0.1 ± 0.124	0.137 ± 0.027	0.032 ± 0.035	0.065 ± 0.146	9.547 ± 1.734	0.374 ± 0.236	1.025 ± 0.209	NA	0.001 ± 0.001	0.025 ± 0.034
	Fedora 32	0.021 ± 0.004	0.009 ± 0.009	0.339 ± 0.106	303.511 ± 0.011	0.09 ± 0.126	0.141 ± 0.023	0.028 ± 0.043	0.005 ± 0.007	11.6 ± 0.638	0.362 ± 0.135	0.988 ± 0.236	0.321 ± NA	0.001 ± 0.002	0.041 ± 0.047
	Ubuntu Focal Fossa	0.022 ± 0.001	0.005 ± 0.005	0.127 ± 0.014	539.857 ± 0.003	0.295 ± 0.049	0.159 ± 0.033	0.018 ± 0.023	0.004 ± 0.004	13.408 ± 1.038	0.447 ± 0.177	1.094 ± 0.185	NA		0.081 ± 0.066
	FreeBSD 12	0.023 ± 0.005	0.008 ± 0.005	0.092 ± 0.01	476.316 ± 0.003	0.06 ± 0.08	0.151 ± 0.037	0.019 ± 0.03	0.003 ± 0.003	8.58 ± 1.673	1.051 ± 0.155	0.828 ± 0.15	0.292 ± NA		0.014 ± 0.025
	MS Win-dows Server	0.022 ± 0.001	0.024 ± 0.006	0.452 ± 0.115	6615.876 ± 1.037	0.233 ± 0.178	0.243 ± 0.033	0.057 ± 0.041	0.012 ± 0.005	25.98 ± 1.927	1.16 ± 0.606	1.429 ± 1.006		0.004 ± 0.006	0.08 ± 0.132
SUS-PEND	Ubuntu Bionic Beaver	0.009 ± 0	0.001 ± 0.002	0.024 ± 0.009	0.003 ± 0.002	0.025 ± 0.055	0 ± 0.001	0.009 ± 0.018	0.003 ± 0.002	1.775 ± 0.556	0.035 ± 0.042	0.156 ± 0.034	NA	0 ± 0	0.009 ± 0.023
	Centos 7 (1300 MB)		0.005 ± 0.006	0.033 ± 0.004		0.04 ± 0.092	0.012 ± 0.02	0.005 ± 0.005	0.004 ± 0.004	2.808 ± 0.327	0.127 ± 0.111	0.185 ± 0.119		0.001 ± 0.002	0.013 ± 0.034
	Centos 7 (898 MB)	0.011 ± 0.007	0.002 ± 0.003	0.026 ± 0.006		0.008 ± 0.005	0.004 ± 0.007	0.08 ± 0.077	0.002 ± 0.003	2.436 ± 0.345	0.059 ± 0.028	0.153 ± 0.021		0 ± 0	0.003 ± 0.003
	Cirros	0.009 ± 0	0.003 ± 0.004	0.014 ± 0.003	0.001 ± 0.001	0.009 ± 0.009	0.012 ± 0.017	0.002 ± 0.001	0.005 ± 0.003	1.599 ± 0.288	0.03 ± 0.008	0.181 ± 0.07		NA	0.007 ± 0.004

Table 3 (continued)

Operation	Image	Ceilom-eter	Cinder	Designate	Glance	Heat	Keystone	Magnum	Manila	MISC	Neutron	Nova	Rab-bitMQ	Sahara	Trove
	Debian 10	0.009 ± 0.001	0.003 ± 0.005	0.021 ± 0.002	0.002 ± 0.002	0.108 ± 0.106	0.002 ± NA	0.008 ± 0.007	0 ± 0	2.237 ± 0.38	0.076 ± 0.126	0.104 ± 0.045		0 ± 0	0.001 ± 0.001
	Fedora 31	0.009 ± 0	0.001 ± 0.003		0.001 ± 0.001	0.03 ± 0.053	0.012 ± 0	0.012 ± 0.021	0.013 ± 0.037	1.694 ± 0.579	0.038 ± 0.041	0.145 ± 0.034			0.002 ± 0.002
	Fedora 32		0.006 ± 0.007	0.108 ± 0.033	0.004 ± 0.005	0.022 ± 0.058	0.006 ± 0.01	0.016 ± 0.025	0.001 ± 0.003	2.937 ± 0.277	0.071 ± 0.085	0.116 ± 0.074		0.001 ± 0.001	0.016 ± 0.031
	Ubuntu Focal Fossa	0.011 ± 0.007	0.003 ± 0.004	0.044 ± 0.013	0.003 ± 0.003	0.153 ± 0.164	0.008 ± 0.014	0.036 ± 0.053	0.003 ± 0.003	3.815 ± 0.513	0.097 ± 0.119	0.155 ± 0.09		0.002 ± 0.002	0.006 ± 0.005
	FreeBSD 12	0.009 ± 0	0.002 ± 0.004	0.013 ± 0.004	0 ± 0	0.006 ± 0.005	0 ± 0	0.002 ± 0.002	0.002 ± 0.002	1.054 ± 0.425	0.174 ± 0.053	0.072 ± 0.025		0 ± 0	0.01 ± 0.021
	MS Windows Server			0.031 ± 0.031	0.003 ± 0.005	0.019 ± 0.05	0.001 ± 0.003	0.016 ± 0.025	0.002 ± 0.003	1.03 ± 0.422	0.085 ± 0.05	0.064 ± 0.015			0.004 ± 0.008
RESUME	Ubuntu Bionic Beaver		0.001 ± 0.004	0.014 ± 0.005	0.001 ± 0.002	0.036 ± 0.065	0 ± 0	0.011 ± 0.021		1.46 ± 0.395	0.041 ± 0.057	0.2 ± 0.041		0.001 ± 0.002	0.01 ± 0.023
	Centos 7 (1300 MB)		0.009 ± 0.01	0.017 ± 0.006		0.088 ± 0.122	NA	0.044 ± 0.079	0 ± 0	1.626 ± 0.22	0.056 ± 0.021	0.207 ± 0.055		0.004 ± 0	0.097 ± 0.125
	Centos 7 (898 MB)	0.009 ± 0.001	0.003 ± 0.005	0.015 ± 0.004	0 ± 0	0.014 ± 0.022	0.001 ± NA	0.093 ± 0.106	0.002 ± 0.003	1.624 ± 0.238	0.093 ± 0.108	0.241 ± 0.11		0 ± 0	0.004 ± 0.005
	Cirros		0 ± 0		0.001 ± 0.001	0.009 ± 0.012	NA	0.032 ± 0.046	0.001 ± 0.002	1.733 ± 0.189	0.032 ± 0.017	0.194 ± 0.019		0 ± NA	0.004 ± 0.003
	Debian 10	0.009 ± 0	0.004 ± 0.004	0.016 ± 0.005	0.005 ± 0.009	0.093 ± 0.003		0.039 ± 0.06	0.006 ± 0.002	1.671 ± 0.147	0.052 ± 0.025	0.128 ± 0.046		0.001 ± 0.002	0.008 ± 0.006
	Fedora 31		0.003 ± 0.004	0.013 ± 0.003	0 ± 0	0.023 ± 0.027	0.006 ± 0.007	0.011 ± 0.02	0.002 ± 0.003	1.383 ± 0.397	0.051 ± 0.047	0.189 ± 0.033		0.001 ± 0.001	0.001 ± 0
	Fedora 32			0.104 ± 0.031	0.002 ± 0.004	0.008 ± 0.026	0.012 ± 0.011	0.012 ± 0.026	0.003 ± 0.003	1.646 ± 0.287	0.033 ± 0.018	0.106 ± 0.037		0.001 ± 0.002	0.004 ± 0.005
	Ubuntu Focal Fossa		0.002 ± 0.003	0.013 ± 0.005	0.002 ± 0.002	0.1 ± 0.013	0.037 ± NA	0.03 ± 0.049	0 ± NA	1.863 ± 0.309	0.035 ± 0.017	0.127 ± 0.087		0 ± NA	0.003 ± 0.003
	FreeBSD 12		0.002 ± 0.004	0.014 ± 0.005	0 ± 0	0.008 ± 0.007	0.02 ± 0.033	0.004 ± 0.003	0.001 ± 0.002	1.345 ± 0.505	0.205 ± 0.05	0.094 ± 0.029		0.001 ± 0.002	0.004 ± 0.006
	MS Windows Server		0.002 ± 0.003	0.033 ± 0.038	0.002 ± 0.004	0.009 ± 0.01	0.013 ± 0.021	0.013 ± 0.024	0 ± 0	1.545 ± 0.345	0.123 ± 0.067	0.1 ± 0.033		0.002 ± 0.002	0.013 ± 0.016

Table 3 (continued)

Operation	Image	Ceilometer	Cinder	Designate	Glance	Heat	Keystone	Magnum	Manila	MISC	Neutron	Nova	RabbitMQ	Sahara	Trove
STOP	Ubuntu Bionic Beaver		0.003 ± 0.004	0.039 ± 0.004	0.002 ± 0.002	0.013 ± 0.011	0.016 ± 0.018	0.006 ± 0.014	0.004 ± 0.003	2.823 ± 0.91	0.053 ± 0.02	0.198 ± 0.068		0.001 ± 0.001	0.014 ± 0.026
	Centos 7 (1300 MB)	0.009 ± 0.001	0.001 ± 0.002	0.016 ± 0.008	0.001 ± 0.001	0.048 ± 0.107	NA	0.021 ± 0.04	0.001 ± 0.002	1.253 ± 0.229	0.046 ± 0.046	0.141 ± 0.059		0 ± 0	0.007 ± 0.01
	Centos 7 (898 MB)	0.009 ± 0	0.004 ± 0.004	0.024 ± 0.003	0.002 ± 0.002	0.008 ± 0.005		0.072 ± 0.107	0.004 ± 0.003	2.195 ± 0.306	0.072 ± 0.129	0.192 ± 0.09			0.002 ± 0.002
	Cirros	0.011 ± 0	0.014 ± 0.005	0.247 ± 0.007	0.02 ± 0.003	0.16 ± 0.144	0.013 ± 0.001	0.085 ± 0.044	0.007 ± 0.004	20.769 ± 0.301	0.451 ± 0.166	0.795 ± 0.105		0.004 ± 0.001	0.08 ± 0.05
	Debian 10	0.011 ± 0.007	0 ± 0	0.028 ± 0.022	0.002 ± 0.002	0.067 ± 0.02	0.009 ± 0.014	0.002 ± 0.001	0.003 ± 0.004	2.086 ± 0.49	0.027 ± 0.012	0.1 ± 0.083		0 ± 0	0.025 ± 0.048
	Fedora 31	0.01 ± 0.004	0.001 ± 0.004	0.021 ± 0.005	0.001 ± 0.001	0.042 ± 0.068	0.009 ± 0.009	0.009 ± 0.02	0.001 ± 0.002	1.607 ± 0.548	0.039 ± 0.043	0.138 ± 0.047		0.001 ± 0.001	0.015 ± 0.027
	Fedora 32	0.009 ± 0	0.004 ± 0.005	0.085 ± 0.027	0.002 ± 0.002	0.038 ± 0.097	0.007 ± 0.007	0.018 ± 0.034		2.161 ± 0.367	0.032 ± 0.012	0.093 ± 0.051		0.002 ± 0.002	0.015 ± 0.034
	Ubuntu Focal Fossa	0.011 ± 0.007	0.004 ± 0.004	0.046 ± 0.015	0.001 ± 0.002	0.074 ± 0.035	0.009 ± 0.014	0.008 ± 0.006		3.94 ± 0.574	0.124 ± 0.144	0.098 ± 0.036		0 ± 0	0.013 ± 0.024
	FreeBSD 12	0.013 ± 0.007	0.014 ± 0.005	0.258 ± 0.009	0.004 ± 0.004	0.1 ± 0.099	0.018 ± 0.023	0.029 ± 0.034	0.007 ± 0.003	15.12 ± 0.663	1.939 ± 0.182	0.482 ± 0.205		0.001 ± 0.001	0.051 ± 0.04
	MS Windows Server	0.015 ± 0.009	0.016 ± 0.006	0.293 ± 0.067	0.018 ± 0.009	0.162 ± 0.126	0.024 ± 0.018	0.034 ± 0.035	0.009 ± 0.004	16.308 ± 0.771	0.664 ± 0.426	0.486 ± 0.123		0.002 ± 0.002	0.054 ± 0.096
SHELVE	Ubuntu Bionic Beaver	0.033 ± 0.002	0.01 ± 0.006	0.151 ± 0.011	1113.557 ± 0.088	0.105 ± 0.099	0.072 ± 0.02	0.024 ± 0.032	0.01 ± 0.012	11.663 ± 1.583	0.307 ± 0.141	0.824 ± 0.114			0.025 ± 0.031
	Centos 7 (1300 MB)	0.035 ± 0.001	0.029 ± 0.028	0.233 ± 0.014	1401.211 ± 0.56	0.26 ± 0.171	0.067 ± 0.01	0.118 ± 0.105	0.008 ± 0.004	20.03 ± 0.874	0.79 ± 0.334	1.423 ± 0.537			0.098 ± 0.151
	Centos 7 (898 MB)	0.034 ± 0.001	0.013 ± 0.013	0.169 ± 0.021	919.581 ± 0.527	0.179 ± 0.134	0.089 ± 0.014	0.077 ± 0.075	0.006 ± 0.003	14.844 ± 0.441	0.462 ± 0.242	0.898 ± 0.133			0.049 ± 0.055
	Cirros	0.033 ± 0.001	0.004 ± 0.005	0.026 ± 0.004	28.721 ± 0.027	0.138 ± 0.14	0.056 ± 0.008	0.003 ± 0.001	0.001 ± 0.002	4.087 ± 0.345	0.118 ± 0.082	0.587 ± 0.052			0.007 ± 0.008
	Debian 10	0.033 ± 0.002	0.018 ± 0.005	0.262 ± 0.021	1512.123 ± 0.106	0.61 ± 0.139	0.062 ± 0.011	0.093 ± 0.052	0.009 ± 0.004	24.048 ± 0.655	0.728 ± 0.214	0.971 ± 0.221		0.003 ± 0.002	0.09 ± 0.054
	Fedora 31	0.033 ± 0.001	0.01 ± 0.008	0.148 ± 0.06	979.696 ± 0.071	0.116 ± 0.113	0.064 ± 0.011	0.014 ± 0.023	0.028 ± 0.057	9.809 ± 1.647	0.269 ± 0.143	0.654 ± 0.1	0.273 ± NA	0.001 ± 0.002	0.014 ± 0.023

Table 3 (continued)

Operation	Image	Ceilometer	Cinder	Designate	Glance	Heat	Keystone	Magnum	Manila	MISC	Neutron	Nova	RabbitMQ	Sahara	Trove
	Fedora 32	0.032 ± 0.006	0.015 ± 0.013	0.383 ± 0.145	836.53 ± 0.118	0.174 ± 0.161	0.07 ± 0.019	0.066 ± 0.053	0.006 ± 0.004	14.959 ± 0.759	0.339 ± 0.159	0.704 ± 0.285	0.264 ± NA	0.002 ± 0.002	0.046 ± 0.049
	Ubuntu Focal Fossa	0.033 ± 0.002	0.019 ± 0.008	0.263 ± 0.012	1365.056 ± 0.087	0.428 ± 0.126	0.081 ± 0.018	0.086 ± 0.061	0.007 ± 0.004	23.964 ± 0.848	0.697 ± 0.108	0.799 ± 0.25	NA		0.13 ± 0.056
	FreeBSD 12	0.034 ± 0.001	0.003 ± 0.004	0.062 ± 0.005	476.908 ± 0.084	0.068 ± 0.093	0.058 ± 0.012	0.018 ± 0.03	0.002 ± 0.003	5.598 ± 1.214	0.669 ± 0.124	0.43 ± 0.122		0.001 ± 0.001	0.007 ± 0.015
	MS Windows Server	0.035 ± 0.001	0.032 ± 0.004	0.659 ± 0.167	6406.869 ± 1208.735	0.319 ± 0.199	0.064 ± 0.016	0.09 ± 0.035	0.018 ± 0.004	37.046 ± 1.885	1.54 ± 0.951	222.094 ± 1209.865		0.004 ± 0.001	0.116 ± 0.197

Table 4 API calls/service (mean ± SD)

Operation	Image	Glance	Keystone	Neutron	Nova
CREATE	Ubuntu Bionic Beaver	4 ± 0	9 ± 1.416	41 ± 1.589	14 ± 0.89
	Centos 7 (1300 MB)		13 ± 1.252	60 ± 1.494	24 ± 0.816
	Centos 7 (898 MB)		11 ± 1.636	47 ± 2.066	18 ± 1.252
	Cirros		10 ± 1.265	32 ± 0.85	10 ± 0.675
	Debian 10		10 ± 1.033	47 ± 2.348	18 ± 0.699
	Fedora 31		10 ± 1.655	41 ± 1.691	14 ± 0.986
	Fedora 32		10 ± 1.042	42 ± 1.524	15 ± 0.855
	Ubuntu Focal Fossa		13 ± 2.003	43 ± 2.394	16 ± 0.738
	FreeBSD 12		11 ± 2.074	39 ± 2.313	13 ± 0.95
	MS Windows Server		14 ± 2.141	100 ± 10.785	47 ± 1.502
SUSPEND	Ubuntu Bionic Beaver	NA	1 ± NA	6 ± 1.143	4 ± 0.484
	Centos 7 (1300 MB)		2 ± 0.707	8 ± 0.568	5 ± 0.422
	Centos 7 (898 MB)		1 ± NA	8 ± 2.685	4 ± 0.316
	Cirros		2 ± NA	5 ± 2.406	3 ± 0.316
	Debian 10		1 ± NA	6 ± 0.699	3 ± 0
	Fedora 31		1 ± 0	6 ± 1.424	4 ± 0.484
	Fedora 32		2 ± 0.707	8 ± 1.39	5 ± 0.379
	Ubuntu Focal Fossa		2 ± NA	12 ± 2.908	6 ± 0.675
	FreeBSD 12		NA	5 ± 2.132	3 ± 0.183
	MS Windows Server		1 ± 0	4 ± 1.455	3 ± 0.254
RESUME	Ubuntu Bionic Beaver		1 ± NA	11 ± 0.774	2 ± 0
	Centos 7 (1300 MB)		NA	11 ± 0.422	
	Centos 7 (898 MB)		1 ± NA	11 ± 0.316	3 ± 0.316
	Cirros		NA	12 ± 2.677	2 ± 0
	Debian 10			11 ± 0.85	3 ± 0.422
	Fedora 31		1 ± 0	11 ± 0.679	3 ± 0.183
	Fedora 32		2 ± 0.548	11 ± 0.466	
	Ubuntu Focal Fossa		3 ± NA	11 ± 0.816	3 ± 0.422
	FreeBSD 12		2 ± NA	11 ± 1.57	3 ± 0.254
	MS Windows Server		2 ± 0.707	11 ± 0.724	3 ± 0.346
STOP	Ubuntu Bionic Beaver		2 ± 0.535	11 ± 2.614	6 ± 0.607
	Centos 7 (1300 MB)		NA	4 ± 0.632	3 ± 0.316
	Centos 7 (898 MB)			8 ± 3.373	4 ± 0.316
	Cirros		2 ± 0.447	65 ± 0.422	30 ± 0
	Debian 10		2 ± 0.707	6 ± 2.53	3 ± 0.667
	Fedora 31		1 ± 0	7 ± 2.811	3 ± 0.403
	Fedora 32			7 ± 2.479	3 ± 0.32
	Ubuntu Focal Fossa		2 ± 0.707	12 ± 2.983	6 ± 0.483
	FreeBSD 12		2 ± 1.166	66 ± 1.213	31 ± 0.669
	MS Windows Server		2 ± 1.029	66 ± 1.474	31 ± 0.583
SHELVE	Ubuntu Bionic Beaver	10 ± 0.183	6 ± 1.383	46 ± 4.071	19 ± 1.159
	Centos 7 (1300 MB)	9 ± 0	6 ± 0.85	66 ± 0.789	27 ± 0.471
	Centos 7 (898 MB)		8 ± 1.269	47 ± 3.498	20 ± 0.919
	Cirros		5 ± 0.516	14 ± 2.547	4 ± 0.422
	Debian 10		6 ± 0.675	74 ± 1.43	31 ± 0.738
	Fedora 31	10 ± 0.183	6 ± 1.184	38 ± 3.845	15 ± 0.615
	Fedora 32		6 ± 1.599	49 ± 3.83	20 ± 0.964
	Ubuntu Focal Fossa	9 ± 0	7 ± 1.524	72 ± 2.716	31 ± 0.738
	FreeBSD 12		5 ± 0.968	23 ± 3.358	8 ± 0.479
	MS Windows Server		6 ± 0.997	150 ± 4.938	66 ± 2.069

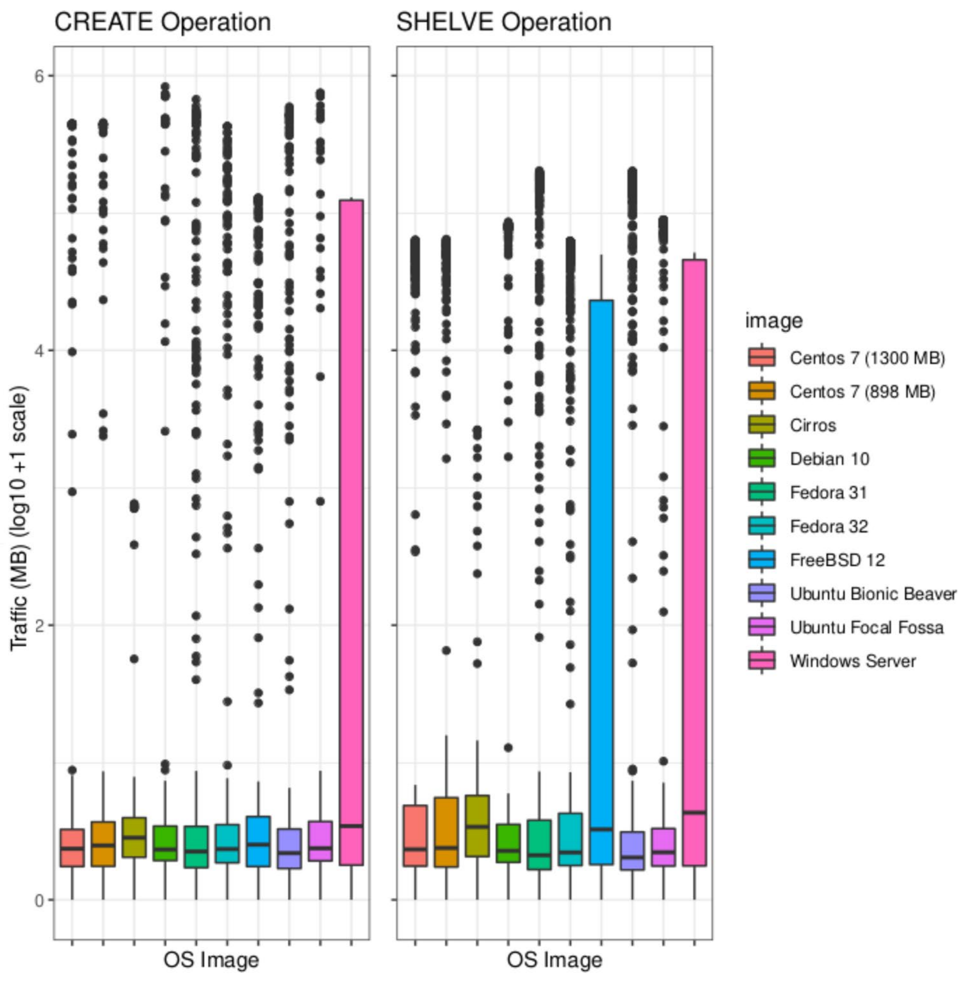


Fig. 8 Box plot of traffic (MB) per second of each OS image

Analysis

Altogether, the experiments are designed to associate the image's size with the resultant network traffic for each operation in the induced lifecycle (Fig. 4). In addition to that, the experiments also allow us to measure the baseline management traffic, which considers only the network traffic strictly necessary for OpenStack to handle the operation requests (excluding tasks such as image transfer through the network). Therefore, such a method is replicable regardless of specific characteristics of the OS images, such as the version. On the other hand, the more images with different sizes, more accurate the results.

From experiment (ii), we confirm that varying the VM flavors does not significantly change the resultant traffic by the operations stated in the induced lifecycle. Such flavors serve as designs for instances created from them, setting configuration parameters such as the number of vCPUs, available RAM, and Disk space. Therefore, such specifications do not significantly affect the network load. However, creating a snapshot from a running VM instance with allocated resources (e.g., memory and disk) could lead to increased traffic in the network, reflecting the snapshot's transmission, which could be investigated in future works.

SUSPEND, RESUME, and STOP operations do not result in heavy network traffic. These operations mostly rely on system calls and tasks performed on the hypervisor. Therefore, the output network traffic depends mainly on the elapsed time. This can be confirmed in Table 2, since different VMs yield a similar amount of traffic per second. On the other hand, CREATE and SHELVE operations produce most significant amount of network traffic. The network traffic measured was classified according to the service it belongs to. Also, the inter-service communication traffic, previously masked as only RabbitMQ traffic, is now mapped into its respective services using the *lsof* tool. 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. Moreover, the number of API calls depends on how the operations are performed (e.g., implementation using Python APIs, CLI). Also, each operation does not require a constant number of API calls. In fact, this may vary depending on configurations, such as the number of network interfaces on the VM instance.

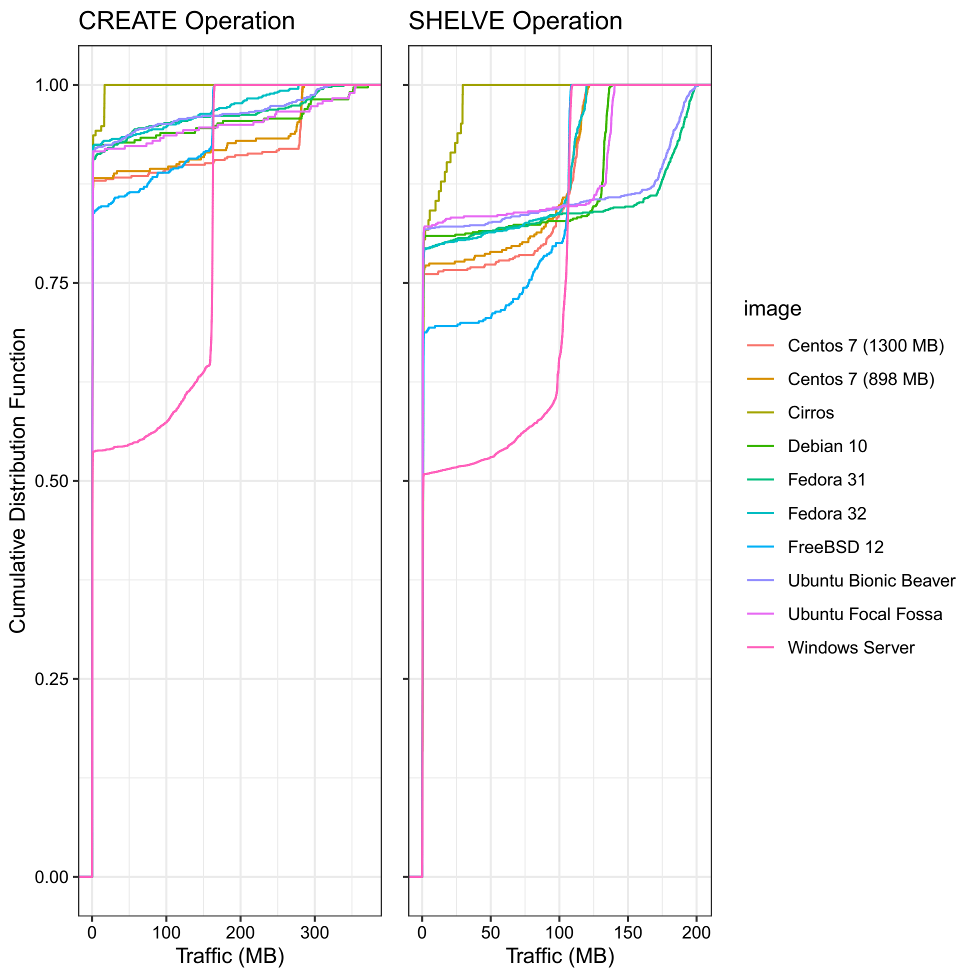


Fig. 9 CDF plot of traffic per second for each OS image

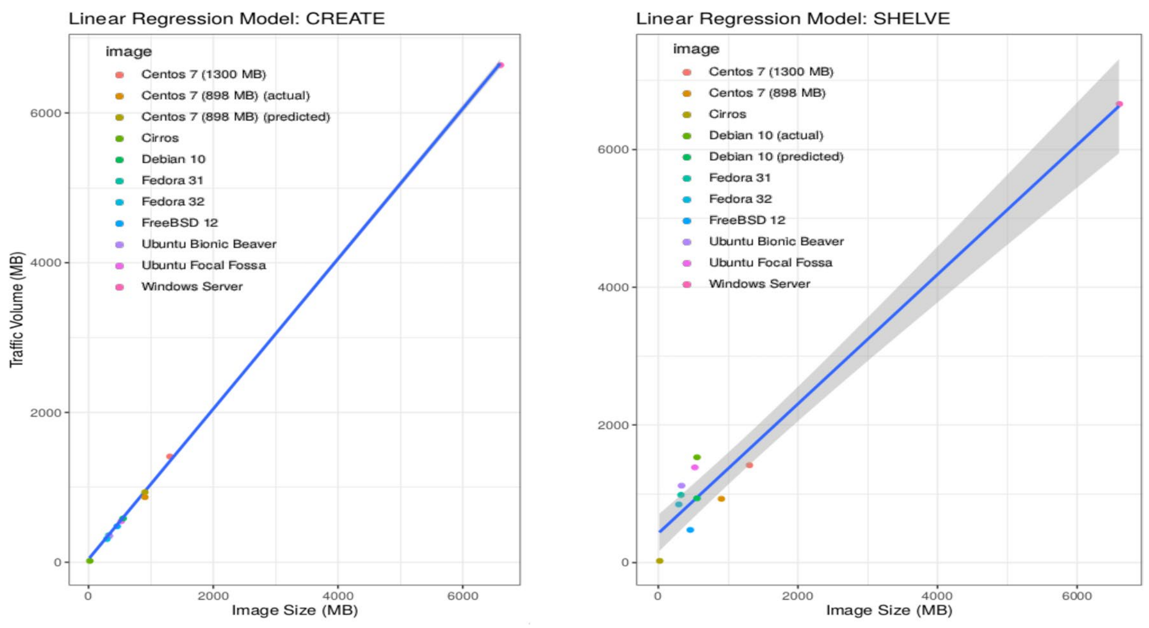


Fig. 10 Linear regression model for CREATE and SHELVE operations. Image size is the predictor and network traffic is the target/response variable

Table 5 Summary of results for Experiment (ii). The network traffic does not change according to the flavor

Image	Operation	Total traffic, MB (mean ± sd)	API calls (mean ± sd)	Execution time, s (mean ± sd)
Ubuntu Bionic Beaver (*m1.small*)	CREATE	358.003 ± 1.663	67 ± 2.9	25.433 ± 1.455
	SUSPEND	2.032 ± 0.586	10 ± 1.464	6 ± 0
	RESUME	1.752 ± 0.436	13 ± 0.819	4 ± 0
	STOP	3.16 ± 0.968	16 ± 3.066	9.8 ± 0.761
	SHELVE	1126.785 ± 1.582	79 ± 5.279	37.767 ± 1.695
Ubuntu Bionic Beaver (*m1.medium*)	CREATE	373.715 ± 1.074	78 ± 4.247	30.7 ± 1.725
	SUSPEND	3.182 ± 0.398	13 ± 2.218	8.1 ± 0.305
	RESUME	2.047 ± 0.219	14 ± 1.813	4 ± 0
	STOP	3.447 ± 0.536	15 ± 3.431	8.567 ± 1.04
	SHELVE	1173.608 ± 1.114	103 ± 5.703	52.933 ± 2.196
Ubuntu Bionic Beaver (*m1.large*)	CREATE	368.394 ± 2.163	68 ± 3.256	24.4 ± 1.192
	SUSPEND	2.19 ± 0.799	13 ± 3.126	7.867 ± 0.507
	RESUME	1.548 ± 0.406	13 ± 1.991	4 ± 0
	STOP	1.976 ± 0.873	12 ± 3.178	7.467 ± 1.383
	SHELVE	1163.87 ± 1.966	81 ± 5.386	38.433 ± 1.524
Ubuntu Bionic Beaver (*m1.xlarge*)	CREATE	368.903 ± 1.613	67 ± 3.604	24.167 ± 1.289
	SUSPEND	2.564 ± 0.77	13 ± 2.091	8 ± 0
	RESUME	2.478 ± 0.508	17 ± 2.95	5.967 ± 0.183
	STOP	2.756 ± 0.791	13 ± 2.891	7.867 ± 1.042
	SHELVE	1172.323 ± 1.356	85 ± 4.241	41 ± 1.174

Creating or shelving VM instances demands a considerable amount of bandwidth, depending on the size of the images. Therefore, content caching is a desirable providence to remain only the management traffic. For instance, such operations may cause the network to clog up if content caching is unavailable and the network is not well designed (e.g., lack of resources, minimal topology). Usually, a content caching joint with a dedicated storage network is the preferred approach. However, such means do not exclude the necessity for resource planning, which avoids under/over resource-provisioning, providing reliability whenever content caching is impossible (e.g., first-time creation, VM instance replica for increased reliability). Therefore, estimating the resultant traffic for VMs creation and shelving are still the most viable approach considering resource planning and network design.

Considerations and Future Work

The present work contributes to a method for characterizing the network traffic in OpenStack's management domain. We perform network monitoring based on some of the most common operations to VM instances, such as creating and stopping them. Our data are available through a summarized dataset published on Zenodo [8]. Additionally, we analyze the impact of such operations on the network and provide a linear regression to predict the resultant network load for creating and shelving instances based on their OS images.

The network traffic characterization allows administrators to understand certain behaviors and plan resource allocation. It is especially challenging to characterize the inter-service communication traffic on OpenStack, since such an inter-service interaction is masked under RabbitMQ traffic. Thus, this paper also presents an alternative to identifying and mapping services communicating over RabbitMQ. Finally, we also point out the possibility for future work on measuring how a snapshot of a working VM with allocated and in-use resources could affect the network traffic and performance.

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

Data availability A summarized dataset was published on Zenodo [8]. Tools were developed in Python 3.6, combining TCPdump for packet collection, and OpenStack Python APIs for handling the VMs: https://github.com/Adnei/openstack_monitor. The tool developed to identify teh traffic by service is available on: https://github.com/Adnei/service_identifier.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Aishwarya K, Sankar S. Traffic analysis using hadoop cloud. In: (ICIIECS), 2015. pp. 1–6. https://doi.org/10.1109/ICIIECS.2015.7192872.
2. Alenezi M, Almustafa K, Meerja KA. Cloud based sdn and nfv architectures for iot infrastructure. Egypt Inf J. 2019;20(1):1–10. https://doi.org/10.1016/j.eij.2018.03.004.
3. Bruneo D. A stochastic model to investigate data center performance and qos in iaas cloud computing systems. IEEE Trans Parallel Distrib Syst. 2014;25(3):560–9. https://doi.org/10.1109/TPDS.2013.67.
4. Chaudhary R, Aujla GS, Kumar N, Rodrigues JJPC. Optimized big data management across multi-cloud data centers: Software-defined-network-based analysis. IEEE Commun Mag. 2018;56(2):118–26. https://doi.org/10.1109/MCOM.2018.1700211.
5. Dainotti A, Pescape A, Claffy KC. Issues and future directions in traffic classification. IEEE Netw. 2012;26(1):35–40.
6. Dainotti A, Pescape A, Ventre G. A packet-level characterization of network traffic. In: 2006 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks, 2006. pp. 38–45. https://doi.org/10.1109/CAMAD.2006.1649716.
7. Donatti A, Miers C, Koslovski G, Pillon M, Carvalho T. Characterization of network management traffic in openstack based on virtual machine state changes. In: Proceedings of the 11th International Conference on Cloud Computing and Services Science - CLOSER. INSTICC, SciTePress, 2021. pp. 232–239. https://doi.org/10.5220/0010447102320239.
8. Donatti AW. Administrative openstack traffic. 2021. https://doi.org/10.5281/zenodo.5785494.
9. Donatti AW, Koslovski GP, Pillon MA, Miers CC. Network traffic characterization in the control network of openstack based on virtual machines state changes. In: Proc. 10th CLOSER, SciTePress, 2020. pp. 347–354. https://doi.org/10.5220/0009404203470354.
10. Finsterbusch M, Richter C, Rocha E, Muller J, Hanssgen K. A survey of payload-based traffic classification approaches. IEEE Comm Surveys Tutor. 2014;16(2):1135–56. https://doi.org/10.1109/SURV.2013.100613.00161.
11. Flittner M, Bauer R. Trex: Tenant-driven network traffic extraction for sdn-based cloud environments. In: 2017 Fourth International Conference on Software Defined Systems (SDS), 2017. pp. 48–53. https://doi.org/10.1109/SDS.2017.7939140.
12. Gustamas RG, Shidik GF. Analysis of network infrastructure performance on cloud computing. In: 2017 International Seminar on Application for Technology of Information and Communication (iSemantic), 2017. pp. 169–174. https://doi.org/10.1109/ISEMANTIC.2017.8251864.
13. Maswood MMS, Medhi D. Optimal connectivity to cloud data centers. In: 2017 IEEE 6th International Conference on Cloud Networking (CloudNet), 2017. pp. 1–6.
14. OpenStack: What to do when things are running slowly. https://docs.openstack.org/operations-guide/ops-maintenance-slow.html. Accessed 13 Aug 2020.
15. OpenStack: Openstack docs: Connection. https://docs.openstack.org/openstacksdk/latest/user/connection.html, 2018. Accessed 15 Jul 2020.
16. OpenStack: Provision an instance, 2018. https://docs.openstack.org/operations-guide/ops-customize-provision-instance.html.
17. OpenStack: Networking architecture, 2019. https://docs.openstack.org/security-guide/networking/architecture.html.
18. OpenStack: Openstack documentation. https://docs.openstack.org, 2019. Accessed 15 Jul 2020.
19. OpenStack: What is openstack? 2019. https://www.openstack.org/software.
20. OpenStack: Message queuing. https://docs.openstack.org/security-guide/messaging.html, 2020. Accessed 22 Jul 2020.
21. OpenStack: Openstack docs: Conceptual architecture. https://docs.openstack.org/install-guide/get-started-conceptual-architecture.html, 2020. Accessed 15 Jul 2020.
22. OpenStack: Virtual machine states and transitions, 2023. https://docs.openstack.org/nova/latest/reference/vm-states.html.
23. Sankari S, Varalakshmi P, Divya B. Network traffic analysis of cloud data centre. In: 2015 International Conference on Computing and Communications Technologies (ICCCT), 2015. pp. 408–413. https://doi.org/10.1109/ICCCT2.2015.7292785.
24. Sciammarella T, Couto RS, Rubinstein MG, Campista MEM, Costa LHMK. Analysis of control traffic in a geo-distributed collaborative cloud. In: 2016 5th IEEE Cloudnet, 2016. pp. 224–229. https://doi.org/10.1109/CloudNet.2016.14.
25. Shete S, Dongre N. Analysis amp; auditing of network traffic in cloud environment. In: 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), 2017. pp. 97–100.
26. Venzano D, Michiardi P. A measurement study of data-intensive network traffic patterns in a private cloud. In: Proc. IEEE/ACM 6th UCC, UCC ’13, IEEE, Washington/DC, USA; 2013. pp. 476–481.
27. Wiki O. Vmstate. 2014. https://wiki.openstack.org/wiki/VMState. Accessed 22 Jul 2020.
28. Williamson C. Internet traffic measurement. IEEE Internet Comput. 2001;5(6):70–4. https://doi.org/10.1109/4236.968834.