sciendo

VOLUME 18, N° 4 2024
Journal of Automation, Mobile Robotics and Intelligent Systems

Comparison of Open Source SDN Controllers and Cloud Platforms in Terms of Performance, Stability, and Infrastructure Flexibility

Submitted: 27th March 2024; accepted: 16th April 2024

Andrzej Mycek

DOI: 10.14313/JAMRIS/4-2024/31

Abstract:
The IT industry is advancing rapidly, with virtually every branch of modern computing experiencing swift development. Concepts such as Cloud Computing and Artificial Intelligence no longer surprise anyone. Recently, Software Defined Networks (SDN) have been gaining significant popularity. This innovative approach to computer networks allows for greater flexibility and is, therefore, much more well-known in the world of cloud computing than in traditional network implementations.
This paper introduces the concept of SDN and Network Functions Virtualization (NFV) and outlines all the challenges and security issues associated with the cloud environment. The dynamic nature of the IT landscape requires constant adaptation to emerging technologies, and SDN represents a noteworthy evolution in the realm of computer networking. Platforms such as SDN and open-source tools enabling the creation of private cloud environments such as OpenStack or OpenNebula were compared. At the same time, aspects like security, network performance, flexibility, and scalability were analyzed. Based on the prior analysis, a comprehensive cloud environment was built using the OpenStack solution and SDN - OpenDaylight was deployed. Additional tests conducted on the OpenStack cloud, both with and without SDN, demonstrated the superiority of SDN implementation in the cloud.

Keywords: *Software-defined network, NFV, cloud computing, OpenDaylight, OpenStack, OpenNebula*

1. Introduction

The concept of Software-Defined Networking (SDN) posits that networks should be designed to be easily managed through specialized software. The foundation of this approach undoubtedly lies in the separation of individual components that communicate with each other through interfaces. Control functions must be centralized on the controller, while the devices are responsible solely for data transmission and executing instructions received from the controller.

This entails a classic division into the Control Plane (controller) and Data Plane (packet forwarding) [11]. Equally important is the Application Plane. On this level, applications are developed, enabling communication and interaction with the entire architecture. The Application Plane is a typical abstract layer [4].

The SDN technology originated from the research project Clean Slate at Stanford University, partly inspired by the OpenFlow protocol. The operational concept of OpenFlow was initially introduced in the paper titled 'OpenFlow: Enabling Innovation in Campus Networks' by Stanford University in 2008, aligning with the idea of Network Functions Virtualization (NFV) [15].

A year later, Professor Nick McKeown formally presented the concept of SDN. In 2012, the American company Google successfully implemented SDN technology in its backbone network, significantly increasing Google's SDN utilization to over 70%. This event marked a breakthrough in the development and future of SDN networks, prompting major global technology companies to introduce their SDN solutions to the market progressively [31].

SDN networks are commonly divided into Underlay and Overlay networks. While these terms may sound mysterious, it is interesting that well-known technologies, such as IPsec and MPLS (Multiprotocol Label Switching), involve Overlay networks. The Overlay layer in SDN networks is where intelligent functions such as segmentation and various policies are implemented. Despite variations in communication protocols at the Overlay layer, depending on the manufacturer, the VXLAN (Virtual Extensible LAN) protocol is commonly used for encapsulation to address scalability issues [31, 32].

On the other hand, the Underlay network is designed for maximum simplicity, serving as a routable IP network. This is why some SDN solutions may involve devices of both older and newer generations, with critical Overlay network functions implemented. The Underlay network has fewer tasks and is intended to provide straightforward routing. Overall, SDN technology offers a nuanced architecture that combines simplicity in the Underlay network with intelligent functionality in the Overlay network.

Additionally, three of the most involved companies in the project, namely Cisco, Juniper Networks, and Extreme Networks, have disclosed their development plans for the NFV architecture proposed by the European Telecommunications Standards Institute (ETSI) [31]. In addition to the IT mentioned above industry giants, there is the OpenDaylight initiative, whose task is to promote implementations. Due to compatibility requirements, OpenDaylight is tailored to the standard SDN ONN architecture along with integration with NFV.

The SDN network's structure in OpenDaylight is divided into three parts:

- Network applications and business processes
- The control platform
- Virtual and physical network devices

OpenDaylight provides a Java Virtual Machine (JVM), and to enhance compatibility with modules from other companies, OpenDaylight also features plug-in modules that allow for additional extension of SDN functionality [5].

2. SDN Network Requirements

The implementation of SDN at an appropriate level involves meeting a series of requirements. As mentioned earlier, one of the most crucial, if not the most critical, requirements is the virtualization of the entire network (overlay and underlay networks). NFV (Network Functions Virtualization) provides both internal and external virtualization. Internal virtualization is used, for example, for virtualizing network devices that serve as network elements. On the other hand, external virtualization connects networks, thus creating a unified virtual network [23]. Equally important is the Application Plane. On this level, applications are developed, enabling communication and interaction with the entire architecture. The Application Plane is a typical abstract layer [32].

SDN networks are also characterized by automation. By eliminating repetitive tasks often performed in traditional networks, we can rapidly create environments and automatically set network parameters related to security or Quality of Service (QoS) policies. Another element that perfectly encapsulates the idea of Software-Defined Networks is programmability. SDN addresses challenges traditional networks struggle with because their concept relies solely on packet exchange between hosts.

However, the vast amount of data transmitted over the Internet today, coupled with the evolution of technologies like Internet Protocol Television (IPTV), video conferencing, Voice over Internet Protocol (VoIP), the rapid increase in network-connected devices, and online gaming, led to the quest for a new paradigm. The programmability of SDN networks makes them resemble mobile applications on smartphones rather than traditional networks. Network Engineers can seamlessly program and manage these networks through Application Programming Interfaces (APIs). Additionally, open APIs facilitate integration with other tools [19].

3. Use of SDN Networks

SDN continues to gain popularity. We previously faced technological gaps in numerous areas, and SDN has been filling these gaps. Currently, the most crucial domain where SDN is successfully utilized is in Data Centers. The history of SDN (Software-Defined Networking) can be traced back to Data Centers. Some even refer to SDN networks as Software-Defined Data Centers (SDDC). This designation is not surprising, given that implementing SDN in Data Centers allows for automation, a revolutionary approach to network management, and the ability to create new environments. SDN is an excellent idea in environments grappling with security issues and complex infrastructure. In homogenous environments, deploying SDN on a large scale may not be practical.

Currently, the most popular commercial SDN solution in the market is undoubtedly Cisco ACI. It serves as an SDN controller, offering a comprehensive set of IP solutions with full integration from the data link layer to the application layer within the SDN network. Additionally, it supports the VXLAN protocol and NFV. One of the advantages of this solution is its capability for cloud deployment, and the tool itself support AWS Cloud. Also highly popular are VMware NSX and Juniper Contrail Networking. An essential distinction between Cisco ACI and VMware NSX is that Cisco ACI is a solution encompassing software and hardware components. Its implementation requires building the infrastructure around Nexus 9000 switches operating in a Spine-Leaf architecture. In contrast, VMware NSX operates exclusively at the overlay layer and does not directly impact the physical network layer. However, an undeniable advantage is that it can function in almost any infrastructure [12].

Among open-source solutions, ONOS SDN controller and OpenDaylight take the lead. In the later part of the paper, OpenDaylight was utilized. This solution is the foundation for commercial controllers like Fujitsu Virtuora and Ericsson SDN Controller. OpenDaylight's strengths include, among others, support for multiple protocols in the Service Abstraction Layer (SBI), encompassing protocols such as Border Gateway Protocol (BGP), OpenFlow, NETCONF, Simple Network Management Protocol (SNMP), and Open vSwitch Database (OVSDB) [2].

Another area where SDN is gaining popularity is campus networks, with users being the primary focus in this case. SDN leverages the technology of numerical control separation to achieve centralized logic management and enhance the programmability of the control plane. This significantly facilitates the simplification of network management. Additionally, it positively impacts performance and security, which is a significant advantage because, as we know, traditional campus networks lack flexibility, making changes challenging and significantly affecting response times in case of failures and associated costs. The third area where the utilization of SDN technology is experiencing significant growth is in Wide Area Networks (WANs).

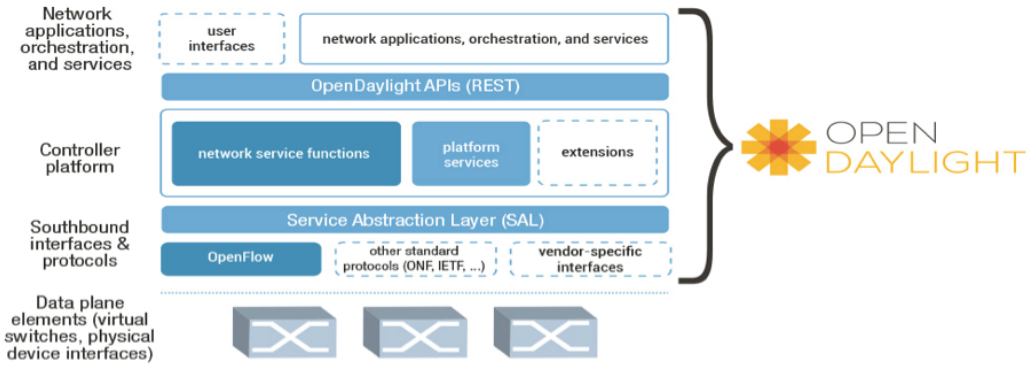

Figure 1. OpenDaylight Architecture [3]

The Software-Defined Wide Area Network (SD-WAN) is particularly well-known in computer networks. This technology enables programmatic control of load balancing and routing, aligning seamlessly with the concept known as Application-driven networking. The expectation from this approach is that the network will cater to the needs of both customers and services, aligning with various applications [27]. The most popular project showing how important this technology is Google's project, which built the B4 structure based on SDN [31].

4. SDN Security and Attack Prevention

Despite its numerous advantages, SDN architecture also comes with security challenges. Beyond the classical threat vectors such as Denial of Service (DoS), data leaks, or unauthorized access, SDN introduces specific threats unique to its architecture. Examples of these threats include attacks targeting the controller software and attempts to disrupt communication between the control plane and the data plane, as well as between the control plane and the application plane [1]. Additionally, controllers with vulnerabilities can cause significant harm to our network. An attacker gaining access to the controller can manage our network by programming switches.

Issues related to Man-in-the-Middle (MITM) attacks are undoubtedly one of SDN's most critical security concerns. A successfully executed MITM attack allows eavesdropping and modifying traffic flow between routers and the network endpoint. Securing the communication channel in SDN involves using the Transport Layer Security (TLS) protocol, which is supported by OpenFlow by default. However, it is essential to note that the TLS protocol and Public Key Infrastructure (PKI) will not protect us if the attacker gains access to the control plane. At this stage, the attacker can easily leverage switches to launch Distributed Denial of Service (DDoS) attacks. Another overlooked point in SDN security is the computers connected to the controller. Computers are often overlooked, and the issue of security is downplayed. In the case of vulnerabilities through computers, gaining access to the controller can be easily accomplished [26].

Despite its incredible capabilities, SDN is unfortunately susceptible to attacks and failures. For instance, in the event of a controller failure, it is advisable to have replicated controllers prepared. It is also recommended to use different controllers, as having identical ones everywhere introduces the risk of encountering the same software bugs across the entire network [13].

Access to the controller must be securely guarded, and communication with the control plane should only be allowed for trusted systems and administrators. In the context of SDN, it is crucial to recognize the AAA method (Authentication, Authorization, and Accounting). Authentication mechanisms such as certificates, passwords, or tokens should be implemented. The authorization process should clearly define the permissions and access to specific resources that an individual has on the control plane.

Critical to security contexts are Intrusion Detection Systems (IDS). IDS is utilized in both traditional infrastructure and cloud infrastructure. The role of IDS is to analyze and identify suspicious activities, promptly notifying IT security teams [17].

Speaking about SDN, we must remember the OpenFlow protocol. Unfortunately, in this case, we encounter additional security challenges. A crucial matter is the protection of the control plane against DoS/DDoS attacks. A specialized framework called AVANT-GUARD has been developed to address such threats in OpenFlow-based networks. This framework consists of two modules and is located in the data plane [28].

5. SDN and NFV in the Cloud

In today's IT industry, two solutions stand out in popularity. Undoubtedly, these are Artificial Intelligence and Cloud Computing. In this discussion, we will focus on the latter, namely cloud computing. As is customary in the IT industry, every branch of modern computing evolves almost daily, and the same holds for cloud computing. The concept of cloud computing is not new—it has been a known model for delivering services over the Internet for several years. Cloud computing offers various services, with the most popular models undoubtedly being Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

In the IaaS model, the service provider makes their infrastructure available to us (such as virtual machines and load balancers). PaaS allows us to deploy our applications using the provider's infrastructure, while SaaS is simply pre-built software from the cloud provider that we can use on the cloud platform [18].

It is inconceivable to envision the operation of even a small organization without leveraging the benefits of cloud computing. However, with the development of this service and the simultaneous increase in its popularity, it becomes more vulnerable to cybercriminal attacks. Hence, one of the primary elements constantly addressed by cloud service providers is improving the overall security of their services. Traditional networks, even in the case of cloud service providers, are now being replaced by SDN to enhance security [23]. Because every second sees a massive amount of traffic generated by cloud clients, not only is security crucial in the case of cloud networks, but so is reliability. Opting for a SDN instead of traditional networks can benefit us as cloud service providers [9]. There are plenty of examples of such advantages, and among the most significant are the "programmable" approach to network management, the ability to experiment with networks without impacting the production environment, the potential for network granularity, and the enhancement of its responsiveness. Traditional clouds are incredibly reliable and secure, but there is and will never be any technology that is 100% secure. The Achilles' heel of clouds can be DoS/DDoS attacks, so to mitigate this threat, it is advisable to abandon traditional networks in favour of SDN, which features an isolated control layer from the data layer [23].

6. SDN Integration in the Cloud and Tests

As for SDN controllers, numerous options are available in the market, but OpenDaylight has been selected for this study. Both OpenDaylight and Floodlight are open-source SDN management tools, providing significant flexibility, which allows them to be used in various environments. Undoubtedly, the advantage of OpenDaylight lies in its complexity, making it suitable for more advanced projects, and the fact that a significantly larger community develops it. In article [24], the authors compared controllers, analyzing network performance, architecture, and QoS. The research was conducted with particular emphasis on delays and data loss in various network topologies and usage scenarios such as cloud computing or multimedia processing. The results of the tests conducted by the authors indicated that in as many as 95% of cases, OpenDaylight outperformed Floodlight's quality.

The authors compared OpenDaylight with Floodlight across three different network topologies, considering three scenarios: one with no cross-traffic, another where nearly half of the bandwidth was cross traffic, and a third where the whole bandwidth was filled with cross traffic.

Table 1. Best controller in different situations [24]

Topology	Load	Latency
Single	Low load	OpenDaylight (2.5 times better)
Single	Mid load	Same
Single	Heavy load	Same
Linear	Low load	OpenDaylight (4.4 times better)
Linear	Mid load	Same
Linear	Heavy load	Floodlight (1.2 times better)
Tree	Low load	OpenDaylight (2.1 times better)
Tree	Mid load	OpenDaylight (4.5 times better)
Tree	Heavy load	Same

In the tests, emphasis was placed on latency, jitter, packet loss, and throughput. The authors' research has shown that OpenDaylight performs significantly better in environments where tree topologies are prevalent, such as in Data Centers and cloud computing environments (see: Table 1).

Hence, the choice fell on the OpenDaylight platform.

A significant factor was also that OpenDaylight is significantly more flexible than Floodlight. Floodlight mainly focuses on the OpenFlow protocol, while OpenDaylight supports OpenFlow, Netconf, and YANG. OpenDaylight is an open-source SDN controller under the auspices of the Linux Foundation. The fact that it is an open-source project is significant, as an open-source product was chosen to implement cloud computing in this work. OpenDaylight is fully vendor-independent and can support various network technologies and devices, providing flexibility. With its modular architecture, it is highly flexible and scalable. Moreover, the tool is implemented in Java, allowing it to be installed on virtually any operating system with a Java Virtual Machine (JVM) [3].

OpenDaylight is, therefore, an excellent choice when it comes to integration with the OpenStack platform. This is particularly important because another dilemma was the cloud platform.

6.1. The Choice Between OpenStack and OpenNebula

Two main options were considered: OpenStack and OpenNebula. These are open-source software designed for creating both private and public clouds.

OpenStack is released under the Apache license and consists of various modules. It originated in 2010 as a collaborative project between NASA and Rackspace Hosting. Over time, the project saw increasing contributions from multiple companies and organizations, including major players such as Red Hat, Huawei, Ericsson, Intel, Dell, T-Mobile, Canonical, Fujitsu, HP, Cisco, and Bloomberg.

Currently, the project is managed by the Open Infrastructure Foundation (formerly known as the OpenStack Foundation), bringing together over 500 companies [10].

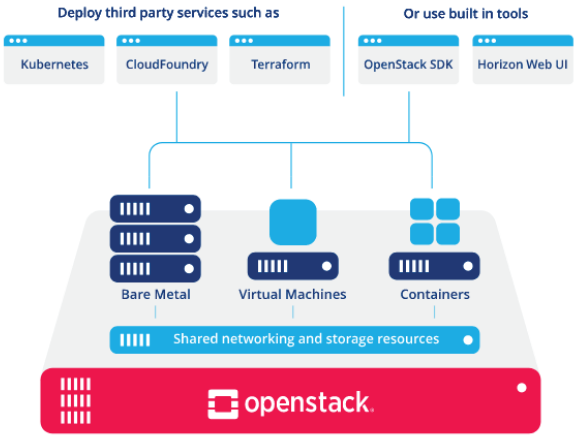

Figure 2. OpenStack Cloud Infrastructure [21]

The latest version of OpenStack, 2023.2 Bobcat, consists of over 40 components and modules, many of which are optional [21].

The key components include:

- **Nova (Compute):** a module for managing virtual machines (VMs) and overseeing the entire lifecycle of instantiated instances;
- **Neutron (Networking):** responsible for networking and connection management between various instances and resources;
- **Cinder (Block Storage):** enables block-level data storage in the cloud, which is crucial for virtual machine instances;
- **Swift (Object Storage):** a module responsible for storing and providing access to objects, such as multimedia files;
- **Glance (Image service):** a module that includes discovering, registering, and retrieving virtual machine images;
- **Keystone (Identity):** facilitates authentication and identity management for all OpenStack components.

Among other popular and available components on the OpenStack platform are, for instance, Ironic (Bare Metal Provisioning Service), Manila (providing file sharing services in the cloud), Octavia (Load balancer), Designate (DNS Service), Barbican (key and certificate management), Heat (Orchestration), Magnum (Container Orchestration Engine Provisioning), Trove (Database as a Service), Horizon (Dashboard), and the modern Skyline (Next generation dashboard) [10, 21].

OpenNebula, much like OpenStack, is also an open-source project enabling the creation of one's cloud computing infrastructure. It was initiated by Spanish researchers Ignacio Martín Llorente and Rubén S. Montero. The project aimed to create a platform well-suited for managing virtual machines in distributed infrastructures.

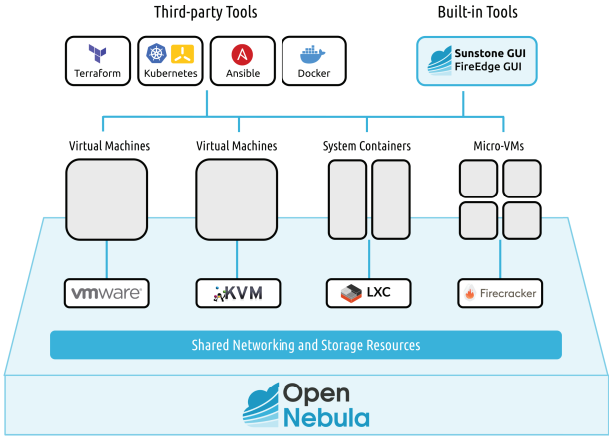

Figure 3. Key features of OpenNebula [20]

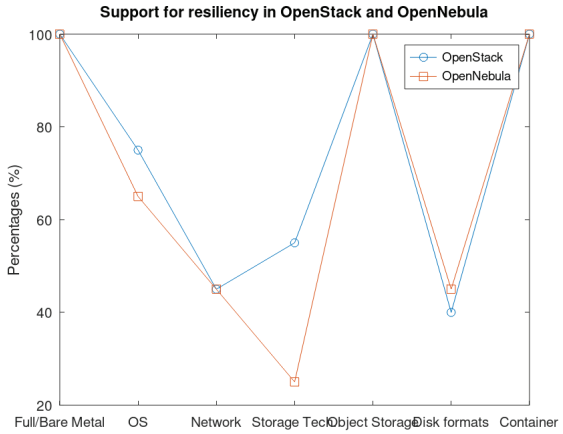

Figure 4. Support for resiliency in OpenStack and OpenNebula [30]

As the project evolved, the authors established C12G Labs, now known as OpenNebula Systems, focusing on designing, consulting, and implementing services based on the OpenNebula solution. The software supports virtualization technologies such as VMware, KVM, LXD, Kubernetes, and Docker. Thanks to a highly user-friendly interface, it allows for easy infrastructure scalability and enables swift and straightforward deployment and delivery of services in the cloud [14, 16].

OpenNebula is a fully platform-independent Enterprise-class product with an open, flexible, and easily extensible architecture, making it suitable for use as a private cloud, public cloud, or in conjunction with another cloud as a hybrid cloud. This platform has been implemented in the most popular programming languages, such as Java, C, C++, and Ruby. It allows centralized management through data storage, monitoring, and network virtualization technologies. Additionally, it allows for integration with OpenCloud, Amazon EC2, OpenStack, or OpenShift [14, 25]. When creating a cloud environment based on OpenNebula, a machine must be configured as the front-end, which will host the cloud manager, while the remaining machines must be configured as slave nodes [29].

Table 2. IaaS tools support for resiliency in OpenStack and OpenNebula [30]

Support	OpenStack	OpenNebula
Full and Bare Metal virtualization	VMware, Hyper-V, KVM, Xen, VirtualBox	VMware, KVM, Xen
OS	RHEL, Fedora, Debian, Ubuntu, SLES	RHEL, Ubuntu, Debian, SLES, CentOS
Network	Neutron, OVS, NSX, PLUMgrid	dummy, ebtables, OVS, VLAN, VMware
Storage	NFS, Ceph, LVM, Gluster, ZFS, Sheepdog	NFS, Ceph
Object Storage	Ceph, Cinder	Ceph
Disk formats	QCOW2, RAw, VHD, LVM, VMDK, VDI	QCOW2, VMFS, LVM, RAW, DEV, Ceph
Container Virtualization	LXC	LXC

In favour of OpenStack compared to OpenNebula, it has a much larger number of functions and popularity, which automatically translates into much better support in terms of the tool's creators and the infrastructure providers themselves. However, the final decision to choose a cloud development tool was made based on [30]'s work. In the above work, the authors compared the flexibility, performance, and resilience of the most popular open-source tools for deploying cloud computing: OpenStack, OpenNebula, and CloudStack. The tests that were conducted showed that OpenStack is the most resilient. To thoroughly investigate these issues, the authors expanded the taxonomy provided in [8]. Since stability is the primary and most important factor in cloud computing, it was the key criterion when choosing a tool to create a cloud computing environment for research on the impact of SDN on cloud computing. System resilience is nothing but the ability to adapt to failures and continuous changes without compromising the level of service availability. This aspect's diversity of infrastructure technologies positively affects the environment's resilience. Table 2 presents a comparison of IaaS tools supporting resiliency. Based on this data, the support for resiliency was calculated considering the number of supported technologies by the cloud creation tool. OpenStack easily outperformed OpenNebula here due to its excellent support for storage, networking, containers, hypervisors, and operating systems (see: Fig. 4).

Another critical criterion when choosing a cloud computing tool was performance. In [30], the authors conducted performance tests of OpenStack, OpenNebula, and CloudStack. The environment consisted of two components: the cloud controller and the node. Performance was calculated based on 40 samples in each micro-test following [22]. The tests were conducted using tools such as LINPACK, STREAM, and IPerf, along with scientific applications [6, 7], on four identical Supermicro blades, each equipped with 24 GB RAM at 1333 MHz, Intel Xeon X5560 (4 Cores, 8 Threads, Max Turbo Frequency 3.20 GHz, Processor Base Frequency 2.80 GHz, Cache 8 MB), and 3 SATA II 7200RPM physical disks in RAID5 configuration connected via a gigabit network [30]. In the performance test under intensive loads, OpenNebula exhibited a poor disk throughput, particularly in write and rewrite operations. OpenStack outperformed the competition in this aspect, and additionally, the results of instances created in OpenStack showed less variability.

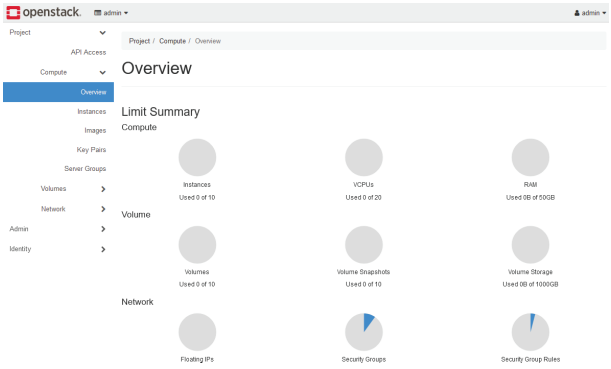

Figure 5. OpenStack Dashboard

Furthermore, in memory tests, OpenStack demonstrated its superiority over competitors. Considering the often-encountered poor performance of instances created in OpenNebula and the fact that OpenStack performed most consistently in the tests conducted by the authors, it was decided to use the OpenStack tool for SDN tests in the cloud.

6.2. Installation and Integration of Tools

The first step was the preparation of the laboratory environment. Open-source cloud computing infrastructure software was installed on a server operating under the control of Ubuntu Server 22.04.3. This software was OpenStack 2024.1 Caracal in the developer version. After a successful installation, basic administrative tasks were carried out, such as adding images with Linux operating systems to the cloud.

Thanks to these actions, we had a preliminary cloud environment ready. To conduct tests, creating a virtual network within the cloud project and deploying instances in it was only necessary. The testing utilized the iperf3 software, an open-source tool commonly used for network testing (bandwidth measurement) [6].

After completing the initial part of the tests (as discussed in the next section), the OpenDaylight Boron software was installed and integrated with the OpenStack cloud to compare the results of tests between the 'pure' cloud and when our cloud is integrated with the SDN platform.

```
$ iperf3 -c 192.168.50.169
Connecting to host 192.168.50.169, port 5201
[  5] local 192.168.50.153 port 45884 connected to 192.168.50.169 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.02   sec  18.4 MBytes   152 Mbits/sec    0    612 KBytes
[  5]   1.02-2.01   sec  14.4 MBytes   121 Mbits/sec    1    612 KBytes
[  5]   2.01-3.01   sec  18.1 MBytes   153 Mbits/sec    0    612 KBytes
[  5]   3.01-4.00   sec  21.1 MBytes   178 Mbits/sec    0    612 KBytes
[  5]   4.00-5.00   sec  15.7 MBytes   132 Mbits/sec    0    612 KBytes
[  5]   5.00-6.02   sec  16.3 MBytes   135 Mbits/sec    0    612 KBytes
[  5]   6.02-7.01   sec  14.6 MBytes   124 Mbits/sec    0    612 KBytes
[  5]   7.01-8.00   sec  20.0 MBytes   168 Mbits/sec    0    612 KBytes
[  5]   8.00-9.00   sec  19.2 MBytes   161 Mbits/sec    0    612 KBytes
[  5]   9.00-10.01  sec  16.7 MBytes   140 Mbits/sec    0    612 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.01  sec   175 MBytes   146 Mbits/sec    1             sender
[  5]   0.00-10.04  sec   174 MBytes   145 Mbits/sec                  receiver

iperf Done.
```

Figure 6. Iperf3 tool

6.3. Tests without SDN

Initially, bandwidth was measured using the iperf tool in the cloud environment (without SDN) for 10 seconds. Subsequently, additional measurements were conducted for 20-second intervals. In the first case, the sender and receiver's average bitrate was 146 Mbits/sec and 145 Mbits/sec, respectively. These numbers were significantly better during the initial 20-second measurement, as the averaged bitrate results indicated a throughput of 4 Mbits/sec higher for both the sender and receiver. However, the second measurement with a 20-second interval yielded decidedly poorer results. A similar situation occurred in the case of measurements with SDN, but that will be discussed in the later part of the article.

The second part of the cloud environment test without SDN involved conducting analogous tests exclusively for the UDP protocol. Two tests, each lasting 20 seconds, were carried out, revealing differences in the delivery time of individual packets averaging 4.773 and 4.757 milliseconds.

6.4. Tests with SDN

To perform this part of the experiment, installing and integrating OpenDaylight with the cloud was necessary. After a successful deployment, tests were conducted using the same methodology, but this time in an SDN environment. The initial 10-second test already demonstrated the superiority of the SDN environment, showing an increase in bitrate by 6 Mbits/sec. In the subsequent two tests with a 20-second interval, the bitrate increase hovered around 15 Mbits/sec.

To conclude the experiment, verifying the results of the UDP protocol was essential. Here, too, the advantage of SDN were evident, with lower jitter values compared to the non-SDN environment.

7. Comparison of the Results

Better results were consistently achieved for OpenStack integrated with OpenDaylight in every test case—regardless of the protocol type, with 10-second and 20-second intervals. In the case of a 10-second interval, SDN-based results were over 4% superior compared to the environment without OpenDaylight.

Table 3. Transfer and throughput measurements for a cloud network without SDN

Type	Interval	Total Transfer	ABR
sender	(0 - 10 sec)	175 MBytes	146 Mbps
receiver	(0 - 10 sec)	174 MBytes	146 Mbps
sender	(0 - 20 sec)	357 MBytes	150 Mbps
receiver	(0 - 20 sec)	357 MBytes	149 Mbps
sender	(0 - 20 sec)	298 MBytes	121 Mbps
receiver	(0 - 20 sec)	298 MBytes	120 Mbps

Table 4. Transfer and throughput measurements for a cloud network with SDN

Type	Interval	Total Transfer	ABR
sender	(0 - 10 sec)	181 MBytes	152 Mbps
receiver	(0 - 10 sec)	181 MBytes	151 Mbps
sender	(0 - 20 sec)	393 MBytes	165 Mbps
receiver	(0 - 20 sec)	393 MBytes	164 Mbps
sender	(0 - 20 sec)	325 MBytes	136 Mbps
receiver	(0 - 20 sec)	324 MBytes	136 Mbps

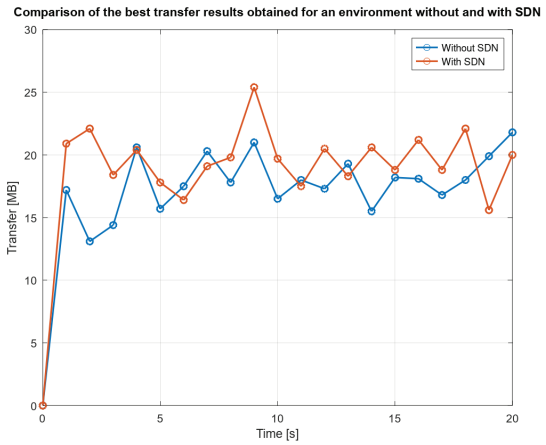

Figure 7. Comparison of the best transfer results obtained for an environment without and with SDN

For 20-second intervals in the traditional environment, the average bitrate for both sender and receiver hovered around 150 Mbits/sec for the first and approximately 120 Mbits/sec for the second tests (see: Table 3).

In the first test scenario for the OpenStack environment with integrated SDN, results were obtained with a bitrate level of 152 Mbits/sec. As mentioned earlier, this is over 4% better than in the case of a traditional cloud environment.

Tests for 20-second intervals also showed that OpenStack with integrated SDN is more efficient and optimal. In the first test of the second test scenario, an average bitrate result of 165 Mbits/sec was achieved (an increase of 10% compared to the traditional environment).

Another test once again demonstrated the advantage of the SDN environment, where a bitrate increase of over 13% was recorded compared to standard OpenStack (see: Table 4).

Table 5. Latency and throughput measurements for UDP in a non-SDN environment

Type	Interval	ABR	Jitter
sender	(0 - 20 sec)	1.05 Mbps	0.000 ms
receiver	(0 - 20 sec)	1.05 Mbps	4.757 ms
sender	(0 - 20 sec)	1.05 Mbps	0.000 ms
receiver	(0 - 20 sec)	1.05 Mbps	4.506 ms

7.1. Tests for the UDP Protocol

The final element of the experiment involved conducting tests for the UDP protocol. Tests using the iperf3 tool in UDP mode help identify network delays and packet loss and are extremely valuable for measuring bandwidth. Such information is beneficial, for example, when providing services related to real-time streaming. In UDP mode, the OpenStack environment with integrated OpenDaylight performed better. In both test cases, jitter (a measure of the temporal variability of delay between packets) was lower in the SDN cloud environment than in the case of the traditional environment. This indicates that OpenStack with integrated SDN allows for more stable data transmission in the network.

Table 6. Latency and throughput measurements for UDP in a non-SDN environment

Type	Interval	ABR	Jitter
sender	(0 - 20 sec)	1.05 Mbps	0.000 ms
receiver	(0 - 20 sec)	1.05 Mbps	4.052 ms
sender	(0 - 20 sec)	1.05 Mbps	0.000 ms
receiver	(0 - 20 sec)	1.05 Mbps	3.769 ms

8. Conclusion

Currently, we are witnessing continuous growth in cloud computing. However, this extends beyond giants like AWS, Azure, or Google Cloud Platform. The same applies to open source projects, which are developing daily with the support of numerous IT enthusiasts and well-known companies. Currently, for our own needs or even for the needs of our organization, we can create our cloud using cloud computing tools such as OpenStack, OpenNebula, Apache CloudStack, or Eucalyptus. Our cloud gives us more customization options and control over its operation.

Additionally, we can influence its performance improvement thanks to various additional mechanisms. This paper demonstrates the benefits and possibilities gained through Software-Defined Networking (SDN) in the OpenStack cloud environment. We can enhance network performance, streamline the management process, and significantly elevate security through integration. SDN can dynamically control traffic, contributing to the optimization of data flow and minimizing delays. The benefits of SDN in the cloud environment were observed through experiments conducted.

It is worth considering how these results will translate in the case of a vast and extensively developed cloud infrastructure. SDN is additionally scalable and flexible, allowing for dynamic management of network resources. Thanks to a centralized approach, it facilitates resource management from a single point. The centralized approach is also an additional advantage in terms of security, making it easier to ensure consistency in network security, which cannot be said for traditional networks, as mentioned at the beginning of the study. With the constant development and popularity of cloud computing, SDN will continue to evolve as the characteristics of this solution are tailored to such large-scale projects.

AUTHOR

Andrzej Mycek* – Cracow University of Technology, Faculty of Computer Science and Telecommunications, Department of Computer Science ul. Warszawska 24, 31-155 Cracow, Poland, e-mail: andrzej.mycek@pk.edu.pl.

*Corresponding author

References

[1] M. Akbaş, E. Karaarslan, and C. Güngör, "A Preliminary Survey on the Security of Software-Defined Networks", *International Journal of Applied Mathematics, Electronics and Computers*, vol. 4, pp. 184–189, 2016.

[2] F. Badaro Neto et al., "SDN Controllers -A Comparative approach to Market Trends", *9th International Workshop on ADVANCEs in ICT Infrastructures and Services (ADVANCE 2021)*, pp. 48–51, 2021.

[3] S. Badotra and J. Singh, "Open Daylight as a Controller for Software Defined Networking", *International Journal of Advanced Computer Research*, vol. 8, pp. 1105–1111, 2017.

[4] M. Barrera Pérez et al., "State of the art in software defined networking (SDN)", *Visión electrónica*, vol. 13, pp. 178–194, 2019.

[5] B. Barros et al., "Applying Software-defined Networks to Cloud Computing", *33rd Brazilian Symposium on Computer Networks and Distributed Systems*, pp. 54–107, 2015.

[6] D. Bourilkov et al., "Using virtual Lustre clients on the WAN for analysis of data from high energy physics experiments", *Journal of Physics: Conference Series*, vol. 396, no. 3, 2012.

[7] J. Dongarra, P. Luszczek, and A. Petitet, "The LINPACK Benchmark: past, present and future", *Concurrency and Computation: Practice and Experience*, vol. 15, pp. 803–820, 2015.

[8] R. Dukaric and M. B. Juric, "Towards a unified taxonomy and architecture of cloud frameworks", *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1196–1210, 2013.

[9] A. Fressancourt and M. Gagnaire, "A SDN-based network architecture for cloud resiliency", *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 479–484, 2015.

[10] D. Grzonka, "The Analysis of OpenStack Cloud Computing Platform: Features and Performance", *Journal of Telecommunications and Information Technology*, vol. 3, no. 3, pp. 52–57, 2015.

[11] A. Hakiri et al., "Software-Defined Networking: Challenges and research opportunities for Future Internet", *Computer Networks*, vol. 75, pp. 453–471, 2014.

[12] P. Ijari, "Comparison between Cisco ACI and VMWARE NSX", *IOSR Journal of Computer Engineering*, vol. 19, pp. 70–72, 2017.

[13] D. Kreutz, F. Ramos, and P. Veríssimo, "Towards secure and dependable software-defined networks", *Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pp. 55–60, 2013.

[14] R. Kumar et al., "OpenNebula: Open Source IaaS Cloud Computing Software Platforms", *National Conference on Computational and Mathematical Sciences (COMPUTATIA-IV)*, 2014.

[15] N. McKeown et al., "OpenFlow: Enabling Innovation in Campus Networks", *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[16] D. Milojičić, I. Llorente, and R. Montero, "OpenNebula: A Cloud Management Tool", *IEEE Internet Computing*, vol. 15, no. 2, pp. 11–14, 2011.

[17] A. Mycek, "Monitoring, Management, and Analysis of Security Aspects of IaaS Environments", *Journal of Telecommunications and Information Technology*, vol. 4, no. 4, pp. 108–116, 2023.

[18] A. Mycek, D. Grzonka, and J. Tchórzewski, "Agent-based Simulation and Analysis of Infrastructure-as-Code Process to Build and Manage Cloud Environment", *ECMS 2023: Proceedings of the 37th ECMS International Conference on Modelling and Simulation*, pp. 513–520, 2023.

[19] B. Omayma and Y. Laaziz, "Software Defined Networks (SDN): the new Era of Networking", *Advanced Information Technology, Services and Systems (AIT2S-15)*, 2015.

[20] OpenNebula. "OpenNebula Docs: 6.8".

[21] OpenStack. "OpenStack Docs: 2023.2".

[22] A. Paradowski, L. Liu, and B. Yuan, "Benchmarking the Performance of OpenStack and CloudStack", *2014 IEEE 17th International Symposium on Object/Component-Oriented Real-Time Distributed Computing*, pp. 405–412, 2014.

[23] P. Patel, V. Tiwari, and M. Abhishek, "SDN and NFV integration in openstack cloud to improve network services and security", *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, pp. 655–660, 2016.

[24] S. Rowshanrad, V. Abdi, and M. Keshtgari, "Performance evaluation of sdn controllers: Floodlight and OpenDaylight", *IIUM Engineering Journal*, vol. 17, no. 11, pp. 47–57, 2016.

[25] G. Rycaj, "Comparison of virtualization performance of Proxmox, OpenVZ, OpenNebula, Vmware ESX and Xen Server", *Journal of Computer Sciences Institute*, vol. 12, pp. 214–219, 2019.

[26] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A Survey of Security in Software Defined Networks", *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 623–654, 2016.

[27] P. Segeč et al., "SD-WAN - architecture, functions and benefits", *2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pp. 593–599, 2020.

[28] S. Shin et al., "AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-Defined Networks", *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pp. 413–424, 2013.

[29] G. Toraldo, *OpenNebula 3 Cloud Computing*, Community experience distilled, Packt Publishing, 2012.

[30] A. Vogel et al., "Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack", *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pp. 672–679, 2016.

[31] Z. Zhang et al., "Software Defined Networking (SDN) Research Review", *Proceedings of the 2018 International Conference on Mechanical, Electronic, Control and Automation Engineering (MECAE 2018)*, pp. 291–300, 2018.

[32] Z. Zhao, F. Hong, and R. Li, "SDN Based VxLAN Optimization in Cloud Computing Networks", *IEEE Access*, vol. 5, pp. 23312–23319, 2017.