Virtual cloud network laboratory based on IaaS with automatized creation of network topology on demand

Yu.A. Ushakov, P.N. Polezhaev, L.V. Legashev, A.E. Shukhman, I.P. Bolodurina
Department of Mathematics and Computer Science
Orenburg State University
Orenburg, Russian Federation
shukhman@gmail.com

***Abstract*—The article describes an approach to create virtual cloud network laboratory. The topology is created with the visual editor. Then a script for network deployment is created. It is based on the scheme description. For implementation of this approach the cloud system based on OpenNebula has been created. The experiment has shown the efficiency of using cloud systems in IT education. Even the small cloud datacenter has provided virtual classes for studying operating systems, network administration, cloud services and creating various cloud services and networks.**

Keywords—virtual laboratory, IT educaton, IaaS.

I. INTRODUCTION

The present day university IT education implies students' access to virtual laboratories. Virtual machines joined in the network are used for practical study of server OS such as Windows and Linux.

The most popular virtualized networking projects (such as GENI, VNET, VNRMS) are considered in the paper [1]. However, we offer a simpler solution for training. It uses a local cloud data center.

Any class requires a certain set of nodes and a topology of their connections. Managed routers and switches used in such networks make access formation complicated. Non-standard topologies and components exclude the use of standard physical schemes. The standard hypervisors (MS Hyper-V, VMware vSphere, Xen, KVM) and cloud aggregators (OpenNebula, OpenStack) provide no possibility to create and deploy the topology on demand using conventional methods.

Cloud services provide the use of any needed number of virtual machines connected to the networks by Open vSwitch (OVS). The use of OpenFlow in cloud infrastructure allows to create any topologies independent from physical constraints. The Open vSwitch is integrated in the topology as a managed switch under the same controller. Nevertheless, such technologies are not supported widely enough as yet.

The system described here is meant for automatized creation and deployment of laboratory and industrial virtual networks with any topology based on OpenNebula cloud platform.

II. PROBLEM DESCRIPTION

The problem of automatized creation and deployment of virtual infrastructure includes creating and editing topology schemes according to current cloud infrastructure possibilities, and converting the schemes into sets of command scripts to deploy infrastructures on demand.

Automatization methods for V-Labs topology creation are presented in some papers. The UNetLab project [3] provides the virtual topology based on single VM. This project uses Qemu, Dynamips and Open vSwitch to emulate real topology and web-based SSH for access to devices. This approach is limited to local memory and CPUs within the single VM. Our project also uses similar Web-based approach for visual topology creation, but we simplified interface for old computer compatibility.

Sometimes, linked virtual nodes must allocate virtual space on different hosts within the common network. Most open source cloud tools use only a real network (VLANs, VXLANs) for segment connection. We have tested GRE based method within the heterogeneous topology [4] to create borderless common network space without any hardware limitation. This method uses virtual GRE links over any real network topology.

A complex management system for both virtual and real networks is presented in our previous paper [5]. This system is based on SDN approach. Within an OpenFlow converged network the controller can make routing decisions without any overhead. In [6] authors use OpenFlow for real-time IP monitoring and routing without MAC address binding. The traffic control method is described in [7]. It works up to L7 and provides protocol-based routing. It can resolve QoS and balancing problem in high-load cloud network to optimal route all the critical network protocols.

The visual design problem with a certain set of components having preset constraints has been considered in a number of researches. The simplest component for use and integration is Draw2D touch component [2] supported in all the browsers. The use of the component allows to store the schemes in JSON format and visually edit them in the browser (Fig. 1).

The description of this example is given in Fig. 2. Due to the simple scheme description and graphic representation, it is possible to build more complex schemes with many interconnected components.

The research has been supported by the Russian Foundation for Basic Research (projects 15-07-06071 and 16-07-01004), and the President of the Russian Federation within the grant for young scientists and PhD students (SP-2179.2015.5).

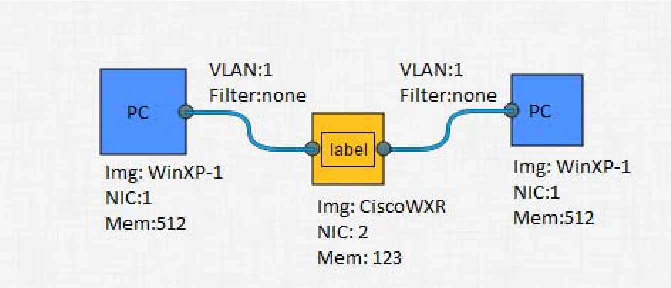

Fig. 1. The example of simple network.


```
{
  "type": "draw2d.shape.basic.Rectangle",
               //Rectangle PC1
  "id": "354fa3b9-a834-0221-2009-abc2d6bd852a",      //ID
  "userData": {IMG:WinXP-1, NIC:1, MEM: 512},        //Load parameters
  "ports": [{                                        //Ports
    "type": "draw2d.Port",                           //Type
    "id": "ebfb35bb-5767-8155-c804-14bd48789dc21",  //ID
    "userData": {name:eth0,speed:100tx,vlan:1},      //Settings
    "name": "eth0",                                  //Name
  }],{
  "type": "draw2d.shape.basic.Rectangle",            //Rectangle PC2
  "id": "354fa3b9-a834-0221-2009-abc2d6bd852a",      //ID
  "userData": {IMG:WinXP-1, NIC:1, MEM: 512},        // Load parameters
  "ports": [{
    "type": "draw2d.Port",                           //Type
    "id": "ebfb35bb-5767-8155-c804-14bd48789dc22",  //ID
     "userData": {name:eth0,speed:100tx,vlan:1},     //Settings
     "name": "eth0",
               //Name
  }],{
  "type": "draw2d.Connection",                       //Link
  "id": "69f7e40c-4586-0d7f-8817-c798f8c07969",
  "userData": {Type:OVS, VLAN:1, Filter:none, Duplex:yes}, //Link settings
  "source": {
   "node": "ebfb35bb-5767-8155-c804-14bd48789dc21"},// Source
  "target": {
   "node": "ebfb35bb-5767-8155-c804-14bd48789dc22"}}//Destination
```


Fig. 2. The JSON description of network topology

The second problem is to create a script for network deployment based on the scheme description. XML RPC is supported in most platforms, and it is used for interaction with OpenNebula. Python extension OCA is used on the server side for the scheme editor and ScriptMaker.

The script to make the infrastructure uses one.template.instantiate() method for creating virtual machines. The information about the parameters of network cards, their number and settings is written in the command of creation. All the virtual machines are created in the “Hold” state. The size of memory and network card parameters can be changed after the virtual machine has been created.

The routing module is used to establish the links between objects based on the object distribution on servers and the network scheme. This module sets up the OpenFlow rules for switches and OVS to provide the topology function. The rules are sets at follows:

<IN_PORT = value> <additional filters> <rewrite rule> <forward port >.

This way to set rules is called pro-active (it implies pre-routed traffic). Each switch and OVS gets its set of rules to distribute and isolate traffic flows. There is no high bandwidth requirements preset, the OpenFlow controller may be used to route all the flows through the rule:

< IN_PORT = value ><forward CONTROLLER>,

which allows dynamic processing of all flows on the controller.

Ryu SDN framework has been chosen as the platform for OpenFlow, due to possibility to configure tables through REST.

III. Implementation Description

For implementation of our approach, the cloud system based on OpenNebula has been created (Fig. 3). All the servers have two network cards: one for the service network and the other for the cloud network. The latter is linked to the OpenFlow switch and by default runs in the regular switch mode with dynamic VLAN.

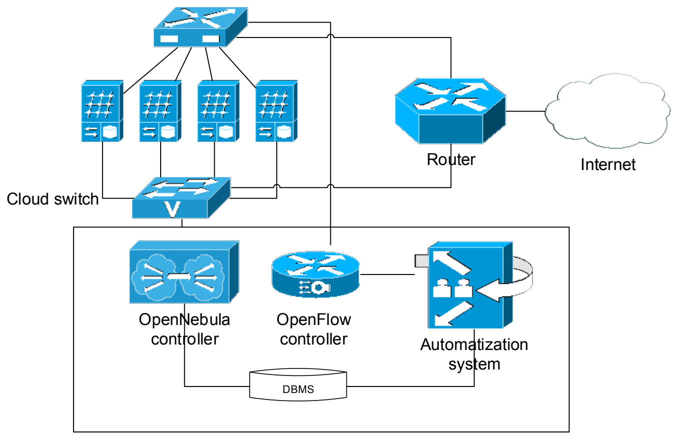

Fig. 3. The scheme of the cloud system

Each server has a virtual switch Open vSwitch (ovs-br0) for connection to virtual machines (Fig. 4). The switch functions as the real one with OpenFlow routing.

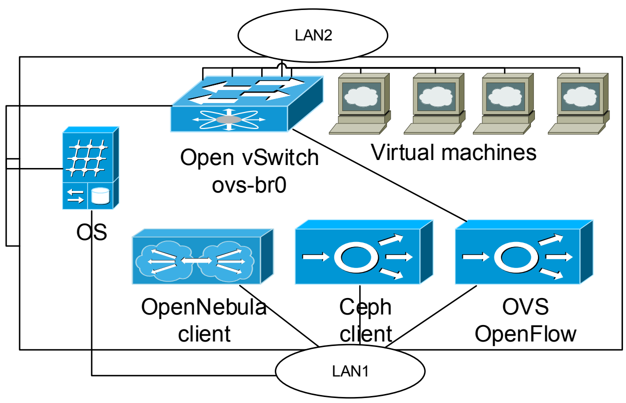

Fig. 4. The JSON description of network topology

From the point of view of controller, each route is a sequence of input and output ports (Fig.5). For routing the Dijkstra algorithm is used on the information, which controller receives through the LLDP protocol.

The network starts after the virtual machines have been deployed and routing rules have been appointed. For access to virtual machine consoles the VNC protocol is used (by default for KVM) through a browser with NoVNC extension.

When viewing the network scheme a double click on a VM opens the window with the console for access to the VM. The access through NoVNC is granted on port 443. Nginx runs on it and forwards traffic to NoVNC. The further access is performed by unique tokens from OpenNebula database, which allows to register real IP addresses and server ports in NoVNC. Each user has access through his tokens only.

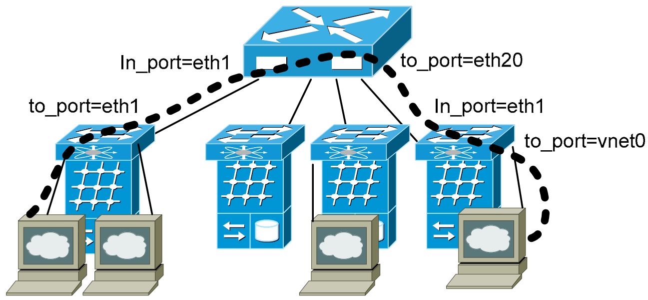

Fig. 5. The route through the network

IV. Access control system

As virtual laboratory infrastructure is to be used for educational purposes, it needs a hierarchy of accounts with distributed access permissions to scheme components.

LDAP has been chosen for account information storage due to its common use. Besides LDAP is integrated with OpenNebula, so there is no need for additional authorization modules. The hierarchy of accounts has the following structure «Administrator» - «Teacher» - «User». Administrators can create or delete any schemes, resources, accounts and virtual machines within their control competence. Teachers can create or delete schemes, accounts, and resources (including virtual machine templates) within their control groups. Users can load, run and stop schemes within quotas granted to them. For automatized start, it is planned to use the class timetable as well as manual start.

V. Experiment description

For our experimental research, we used the university educational cloud data center. The center includes four two-processor servers and eight one-processor servers with 32GB RAM each. Both the service network and the cloud network are implemented by HP 3500 switches supporting OpenFlow. The cloud is hypervised by OpenNebula, all the servers have the Open vSwitch bridge with OpenFlow for interconnection between the network and virtual machines. The Ceph four-server distributed storage is used. Each server has three 1Tb disks with RAM cashing. In addition, KVM save cashing is used for better speed. All the networks function at 1Gb/s, the links between switches provide 10Gb/s.

The first experiment was carried out to test storage performance for different configurations, determining the optimal choice. The virtual machines were started in automatic mode, starting times and Ceph storage latency were measured simultaneously. The start was performed with “onevm resume” command, the running time was registered with the monitoring script based on “onevm list” command. Fig.6 shows the experiment results for 1 through 50 virtual machines.

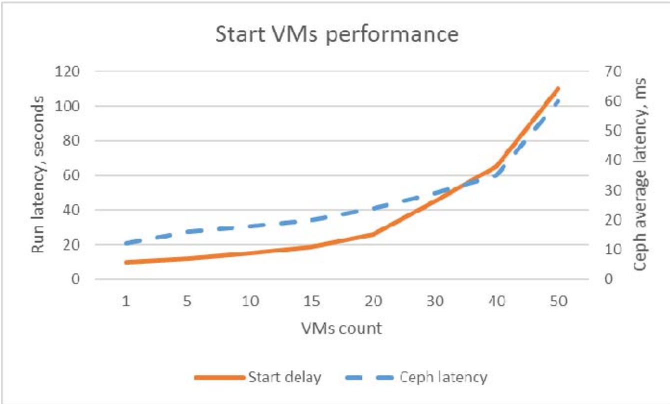

Fig. 6. The start VMs performance

As can be seen in Fig.6 run latency grows exponentially as VMs count grows. It is related to disk storage latency growing. The cloud implies 8 disks on 4 servers that can provide theoretical read performance up to 800Mb/s.

Fig.7 shows the storage throughput at different numbers of read/write processes.

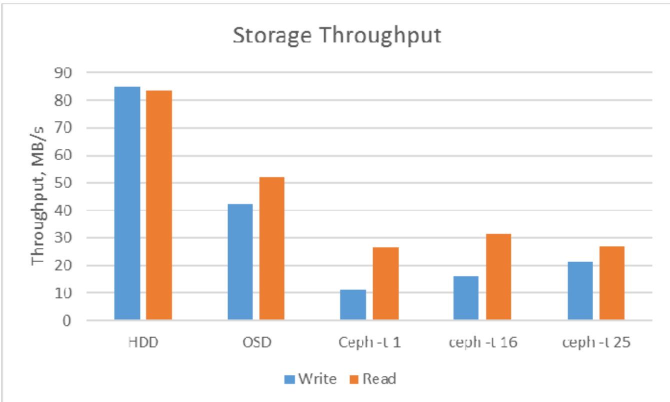

Fig. 7. The storage throughput.

As can be seen the storage throughput is significantly depends on access method. The experiment embraced only one client with several flows. If the number of clients is growing, the read performance grows proportionally, but the write performance remains the same (Fig.8).

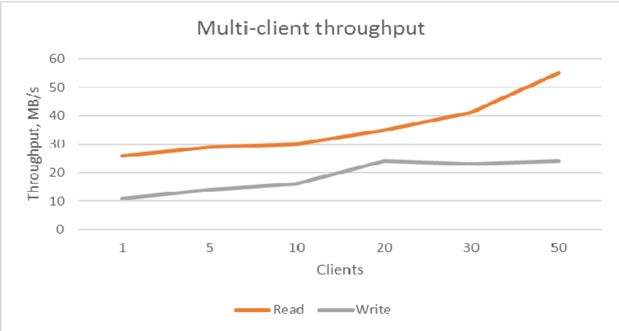

Fig. 8. The multi-client throughput.

This explained by the distributed read method and the logging write method. The use of SSD for cashing and logging may provide a several times better performance.

VI. CONCLUSION

The experiment has shown the efficiency of using cloud systems in IT education. Even the small cloud datacenter (12 low-performance servers) has provided virtual classes for studying operating systems, network administration, cloud services and creating various cloud services and networks.

We are planning to carry on further research using a high-speed storage network with a 10G switch and SSD for cashing.

REFERENCES

[1] Chowdhury, N.M.M.K.; Boutaba, R. Network virtualization: state of the art and research challenge //IEEE Communications Magazine, 2009– vol. 47– n.7– PP. 20-26.

[2] Draw2D touch URL: http://www.draw2d.org/draw2d/Konnov A.L., Legashev L.V., Polezhaev P.N., Shukhman A.E. Concept of Cloud Educational Resource Datacenters for Remote Access to Software // Proceedings of 11th International Conference on Remote Engineering and Virtual Instrumentation (REV), Polytechnic of Porto (ISEP) in Porto, Portugal from 26-28 February 2014 – 2014 – pp-246-247.

[3] UNetLan Official Site. Documentation http://unetlab.com/docs/

[4] Polezhaev P., Shukhman A., Ushakov Yu. Network Resource Control System for HPC based on SDN // Proceedings of 14th International Conference, NEW2AN 2014 and 7th Conference ruSMART 2014, St. Petersburg, Russia. Lecture Notes in Computer Science. – vol. 8638. – PP. 219-230.

[5] Polezhaev P., Shukhman A., Konnov A. Development of educational resource datacenters based on software defined networks // Proceedings of 2014 International Science and Technology Conference “Modern Networking Technologies (MoNeTec)”, Moscow, Russia, 2014. – PP. 133-139.

[6] Stabler G., Rosen A., Goasguen S., Kuang-Ching Wang Elastic IP and security groups implementation using OpenFlow. // VTDC '12 Proceedings of the 6th international workshop on Virtualization Technologies in Distributed Computing Date - PP. 53-60.

[7] Gorja P., Kurapati R. Extending open vSwitch to L4-L7 service aware OpenFlow switch. // 2014 IEEE International Advance Computing Conference (IACC) - PP. 343-347.