Design Engineering

ISSN: 0011-9342
Issue: 1 | Pages: 390 - 397

Construction of Virtual Network Mapping System Based on Openstack Cloud Computing Platform

Dongxia Yang*

Faculty of Software and Big Data,Inner Mongolia Electronic Information Vocational Technical College, Hohhot, Inner Mongolia, 010051, China

*Corresponding Author: Dongxia Yang

Abstract:

The purpose of this paper is to find the implementation of open source cloud computing platform based on Open Stack. The open service is provided in Open Stack for both public and private clouds. In terms of infrastructure as a service (IaaS) platform, the Open Stack cloud platform faces a resource allocation limitation problem. This paper involves a customization of Open Stack Dashboard (horizon) to generate virtual network requests and then processing these requests. It implements a proposed virtual network embedded problem algorithm (VNE-Greedy) on top of the Open Stack Cloud platform. This paper also includes extended research into the Open Stack cloud computing system and its associated virtual network embedding strategies. The results are valuable for the construction of virtual network mapping system.

Keywords*: Evaluation model, Implementation, Open source, Cloud computing platform, Open Stack.*

I. INTRODUCTION

The emergence of cloud computing has changed the traditional IT service mode [1]. Through virtualization technology, cloud computing will make a large number of physical resources into a virtual resource pool. Users can obtain computing, storage, network and other resource services anytime, anywhere. Because of its flexible management, easy expansion and elastic computing, more and more enterprises are gradually migrating their services to cloud computing platform, which not only improves the utilization rate of hardware resources, but also reduces the maintenance cost of enterprises [2-3].

Openstack is the most influential cloud computing management tool [4]. The tool manages the IAAs cloud resource pool (server, storage and network) through commands or web-based

ISSN: 0011-9342
Issue: 1 | Pages: 390 - 397

visual control panel. It was first developed jointly by NASA and Rackspace in 2010 [5]. Now, more than 9500 individuals from more than 100 countries and more than 850 well-known enterprises in the world, such as NASA, Google, HP, Intel, IBM and Microsoft, are involved [6-7].

With the development of cloud computing and the popularity of cloud services, the heterogeneity of cloud data center resources and the diversity of cloud users' demand for resources have brought great challenges to the resource management of cloud platform. How to reasonably allocate resources to users in the complex cloud environment is an important topic of cloud computing technology research. The virtual resources of cloud computing center are mainly organized and distributed in the form of virtual machines. Therefore, the scheduling of virtual machines has become the core of the research. Virtual machine scheduling has an impact on resource utilization, system energy consumption, load balancing and user quality of service. Therefore, the scheduling strategy of virtual machine needs to consider many optimization objectives. Virtual machine scheduling can be divided into two aspects: virtual machine allocation and virtual machine migration. The allocation of virtual machines can be described as the association mapping between virtual machines and physical machines. How to find the best mapping between multiple virtual machines and multiple physical machines under the premise of ensuring multi-faceted optimization objectives is a problem to be solved in the virtual machine allocation strategy. Aiming at the optimization objectives of physical resource balance and system load balance, this paper establishes physical resource surplus model and system load balance model, and designs a virtual machine allocation strategy based on particle swarm optimization algorithm [8]. Fig 1 showsthe virtual network embedding.

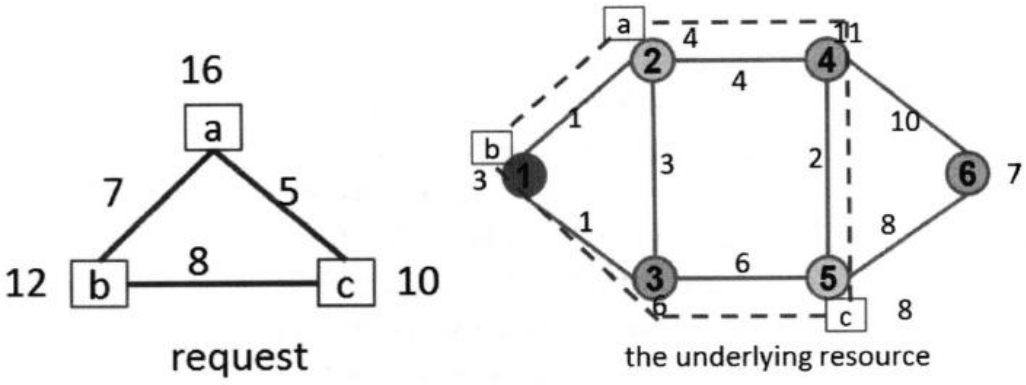

Fig 1: Virtual Network Embedding

Design Engineering
ISSN: 0011-9342
Issue: 1 | Pages: 390 - 397

II. OPEN STACK MODEL AND ALGORITHM

Open stack compute architecture is shown in Fig 2.

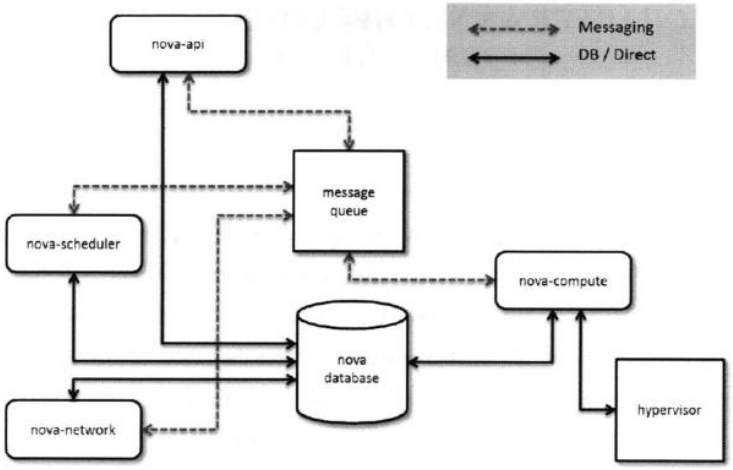

Fig 2: Open Stack Compute architecture overview

Most of the mappings are only node and connection mappings, without considering the topology characteristics, and assuming that the physical device always works well. However, this is not the case, which can lead to serious problems and can not serve the tenants. In this paper, the survivability mapping of virtual network based on topological characteristics is considered, and the different importance of different nodes and connections is considered. The multi controller to switch concurrent connection, path diversity and network delay are used to map to achieve the purpose of survivability characteristics.

The algorithm is divided into two steps, node mapping and connection mapping, and the relationship between them is considered. When considering the VNE survivability, the traditional algorithm does not consider the virtual controller mapping, but treats each node equally, that is to say, it does not combine the network topology and the controller path.

The objective maximization control algorithm used in this paper can ensure the reliability of the network and maintain the success rate and profit ratio.

CPU is used as node weight, and grade and proximity are selected as the optimization strategy of network centrality.

There are two stages: 1. Mapping the virtual SDN controller and the virtual request switch to the physical node, and then decide whether to connect the virtual connection or not. Virtual controller mapping is similar to the general controller location selection problem. The difference lies in logical independence and dynamic change. 2. Virtual connection mapping.

OLSFis used to select the controller location and sort the physical nodes according to the OLSF.The largest OLSF mapping controller is selected [9-10]:

ISSN: 0011-9342
Issue: 1 | Pages: 390 - 397

$$(N, sk) \leftarrow Key(1^k) \quad (1)$$

The virtual request nodes are sorted and mapped to the largest RDC node:

$$\begin{aligned} &r \leftarrow \{0,1\}^k; sk \leftarrow \{e, d, r\}; \\ &Output\{N, sk\}; \end{aligned} \quad (2)$$

The function can be described as:

$$\phi(N) = (p-1)(q-1) \quad (3)$$

Then the the largest RDC node is set to satisfy the following equation 4:

$$\begin{cases} 1<e<\phi(N) \\ \gcd(e, \phi(N))=1 \end{cases} \quad (4)$$

Considering the virtual population topology, RDC is used to select the controller location:

$$(T_0, T_2, ...T_{n-1}) \leftarrow Tag(pk, sk, m) \quad (5)$$

Then controller location can be noted as.

$$for(j = 0; j \le n-1; j++); \quad (6)$$

$$\begin{aligned} &\{W_j = r*(j+1); T_i \\ &= [h(W_j)*m_j]^c \bmod N\}; \end{aligned} \quad (7)$$

$$Output(T_0, T_2, ...T_{n-1}); \quad (8)$$

The requests are sorted according to the revenue, and the controller and virtual switch are mapped. The following Eq.9 is used to select the location of the virtual switch:

$$\begin{aligned} {}_a I_b^{(\alpha)} f(t) &= \frac{1}{\Gamma(1+\alpha)} \int_a^b f(t)(dt)^{\alpha} \\ &= \frac{1}{\Gamma(1+\alpha)} \lim_{\Delta t \to 0} \sum_{j=0}^{j=N-1} f(t_j)(\Delta t_j)^{\alpha} \end{aligned} \quad (9)$$

In which,

$$T(\nabla) = \begin{Vmatrix} T_{ik}(\nabla) & t_i(\nabla) \\ t_k^T(\nabla) & -\tau(\nabla) \end{Vmatrix}, \quad \mathrm{J} = \begin{Vmatrix} \delta_{ik} & 0 \\ 0 & 0 \end{Vmatrix},$$

$$f(x, \omega) = \begin{Vmatrix} u_k(x, \omega) \\ \varphi(x, \omega) \end{Vmatrix} \quad (10)$$

Then we have equation (11) to (12):

ISSN: 0011-9342
Issue: 1 | Pages: 390 - 397

$$C^1 = C - C^0, e^1 = e - e^0 \quad (11)$$

$$\eta^1 = \eta - \eta^0, \rho_1 = \rho - \rho_0 \quad (12)$$

CO-vSDNE and TO-vSDNE are used for virtual requests with constraints on latency, and MC-vSDNM is used for unconstrained requests. The difference is that the formula for calculating the fitness value of physical nodes to be mapped is different, as shown in following equation (13-14):

$$G_{ik}(\bar{k},\omega) = \frac{1}{\rho_0\omega^2}[\frac{\beta^2}{\bar{k}^2-\beta^2}\theta_{ik} + \bar{k}_i\bar{k}_k\left(\frac{1}{\bar{k}^2-\alpha^2}-\frac{1}{\bar{k}^2-\beta^2}\right)+m_i m_k\frac{\beta_\perp^2}{\bar{k}^2-\beta_\perp^2}] \quad (13)$$

$$\mathcal{g}_{ik}(\bar{k},\omega) = -\frac{1}{\eta_{11}^0}\frac{1}{\bar{k}^2}+\frac{1}{\rho_0\omega^2}\left(\frac{e_{15}^0}{\eta_{11}^0}\right)^2\frac{\beta_\perp^2}{\bar{k}^2-\beta_\perp^2} \quad (14)$$

$$\gamma_i(\bar{k}_i,\omega) = \frac{1}{\rho_0\omega^2}\left(\frac{e_{15}^0}{\eta_{11}^0}\right)^2\frac{\beta_\perp^2}{\bar{k}^2-\beta_\perp^2}m_i \quad (15)$$

III. VIRTUAL MACHINE MONITORING ON OPENSTACK

The system monitoring of an operating system is very important in the whole system. For example, in openstack system, the virtual machine is monitored and controlled by Nagiso. Nagios is the real-time monitoring of the host and server under the premise of setting, and even sending out warnings. This is suitable for application layer software for monitoring system or network. It has some unique characteristics, such as:

(1) The system monitors network services (SMTP, POP3, HTTP, NNTP, Ping, etc.), monitors host resources (including processor load, disk utilization, etc.);

(2) The system has the function of parallel service checking mechanism;

(3) The system has plug-in design, which can be very convenient for users to extend self-service detection methods;

(4) The system has the function of defining the network hierarchical structure: by using the "parent" host to express the relationship between the network hosts, it is easy to detect the reachable or unreachable state of the host downtime. Once the service or host has a problem, or the problem has been solved, the alarm will be reported to the contact through email, SMS, user-defined and other means.

(5) The system can define event handle and get more problem location when some event occurs in host / service;

ISSN: 0011-9342
Issue: 1 | Pages: 390 - 397

(6) The system has the function of automatic log rollback;

(7) The system can support and realize the host monitoring;

(8) At this moment, you can view the history of the network, system log, and so on. During the test operation, Nagiso3 is installed by default. The virtual machine configuration is monitored: At the beginning, the command to view the virtual machine status is used at this moment. The system entersNvalist to observe the virtual machine interface display. Fig 3 shows us the formatted VN information and Fig 4 shows the virtual bridge status in the data transporting process.

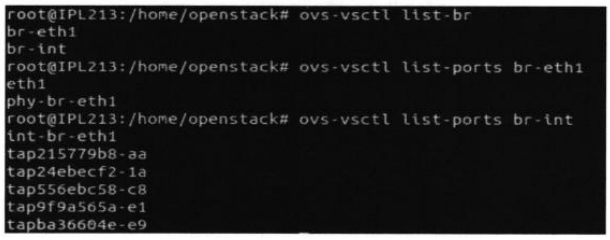

Fig 3: Formatted VN information

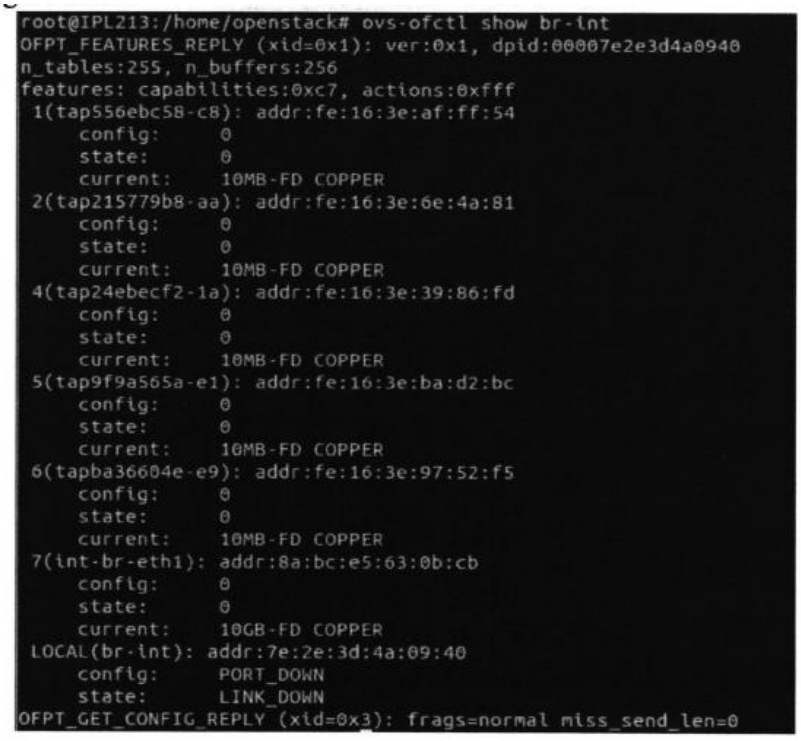

Fig 4: The virtual bridge status

Through the filtering mechanism, you can get a list of all the hosts that meet the conditions. There may be multiple hosts that meet the conditions, so you need to select the best host from them. Openstack provides a trade-off mechanism. The role of trade-off is to calculate and sort

ISSN: 0011-9342
Issue: 1 | Pages: 390 - 397

the weights of the hosts that meet the conditions, and finally select the host with the highest weight as the best host. Openstack calculates the trade-off of available nodes through cost calculation function and corresponding weight factors. First, each filtered host calls the cost calculation function for calculation, and multiplies the calculated result with the corresponding weight factor, and then sums the multiplied results to get the weight value of the host. The host with the smallest weight value is the best choice.Fig 5 shows the Iperf bandwidth test.

Fig 5:Iperf bandwidth test

In openstack, the default cost calculation function is simply to return the remaining memory size. The default value of weight factor is - 1.0, and the product of the two is the cost value of a host. Because the trade-off factor is negative, it means that openstack will try to select the physical host with more memory during the scheduling process. If the trade-off factor is certificate, it means that all virtual machines should be put on the same physical machine as far as possible during the scheduling process. Fig 6 shows the Iperf client side bandwidth test.

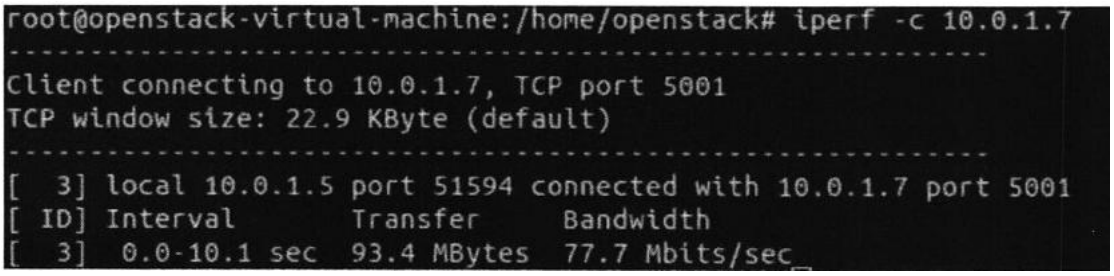

Fig 6:Iperf client side bandwidth test

It can be seen that different cost calculation corresponds to different scheduling strategies, so as to achieve different optimization objectives.

IV. CONCLUSION

Through this paper, our objective was to implement, evaluate and demonstrate of VNE strategy in Open Stack. And that was done by first conducting research on related cloud computing and virtualization moreover by gaining an understanding of the Open-Stack Operating system more specifically Dashboard (Horizon) and Compute (Nova) architecture and its scheduler component. Later, a working test bed built using Open-Stack was set up and used to implement a scheduler for initial VMs nodes. Finally, the test bed was used to validate and conduct experimental evaluation to showcase its performance.

ISSN: 0011-9342
Issue: 1 | Pages: 390 - 397

Customization and modification of the Open Stack Horizon, by customizing the GUI interface for the users to input the nodes and links intend to be embedded in Open Stack, intensive reading and comprehends for Open Stack source code was required along with a good commend of Python and Django to accomplish this task (Customizing Open Stack Dashboard), we also used debugging tools such PDB to capture the data flow and parameters handling in the fabric of the Open Stack.

REFERENCES

[1]Liu, B. J. (2005) Consideration on improving the efficiency in open-stack borrowing. Journal of Baoding Teachers College35(6): 350-5

[2] Chen Xiaohua, Li Chunzhi, Chen Liangyu, Zeng Zhenbing. (2014) Energy efficient virtual network mapping for active sleep node links. Acta software07: 1416-1431

[3] Li Xiaoling, Guo Changguo, Li Xiaoyong, Wang Huaimin. (2012) A constrained optimization based virtual network mapping method. Computer research and development 08: 1601-1610

[4] Wei Xiaohui, Zou Lei,Li Hongliang. (2013) Virtual network mapping algorithm based on optimized isomorphic subgraph search. Journal of Jilin University 01:165-171

[5] Li-Yao, L. I., You, Y., Zhao, S. K., Hua-Rong, X. U. (2015) Dynamic bandwidth allocation strategy based on open stack cloud platform. Journal of Minnan Normal University

[6] Rui, K. (2015) Research on cloud computing system based on open stack platform. Journal of Chongqing University of Science and Technology

[7] Ding Jian, Liu Jiang,Liu Yunjie. (2015) A virtual network mapping algorithm for multi topology type requests. Journal of Beijing University of Posts and Telecommunications03: 88-93

[8] Chang Lei, Gu Huaxi, Zhang Zhiyi, Yu Xiaoshan,Zhao Yan. (2015) A user priority virtual network mapping algorithm based on particle swarm optimization. Journal of Xi'an University of Electronic Science and Technology (NATURAL SCIENCE EDITION)1: 16-22

[9] Yuan Ying, Wang Cong, Wang Cuirong, Song Xin, LV Yanxia. (2016) Virtual network mapping algorithm for dynamic virtual network request. Computer Applications 1 :6-11

[10] Nirmal, D. A., Arunkumar, K., Girish, V., Arunkumar, R. (2014) Setting up based private cloud using open stack. Networking and Communication Engineering