Elastic Resource Provisioning System Based on OpenStack Cloud Platform

Zheng Zhang(✉), Hao Xu, Ke Chen, and Pingping Shan

College of Software, Nanyang Institute of Technology, Nanyang 473000, Henan, China
sawest@163.com

Abstract. As open source private cloud platform, OpenStack provides basic resource service which has features of stability reliability and security. Function components of OpenStack platform were deployed and installed. Integrated basic resource pool was built. Based on open API of platform, applying JAVA native interface, elastic resource provisioning system was implemented which adops software architecture of B/S. The system has functions of establishment distribution and real-time adjustment of dynamic resource. In order to manage elastic resource of plateform expediently the system provides personalization module and favorable user interface.

Keywords: Elastic resource · Cloud platform · OpenStack · B/S · JNI

1 Introduction to OpenStack

OpenStack is open-source software which provides a basic platform for deployment cloud facilitates the deployment and management of virtual machine and serves as a public/private cloud for virtual computing and storage. OpenStack mainly include Nova virtual computing service, Swift storage service, Glance virtual image registration distribution service [1–4].

1.1 System Structure of OpenStack

OpenStack cloud platform mainly includes seven components with different functions:

Nova computing component is the core of OpenStack platform and responsible for computing part and implementing corresponding strategies. Other components are scheduled through Nova component. Same as Amazon EC2 and RackspaceCloud-Servers, Nova component provides functions, including example operation, network management, user control and access to the cloud by other items.

Swift object storage component is a distributed object storage component with similar functions to Hadoop. In addition, Swift is used to store the image files to create virtual machine.

Cinder block storage component adds enduring storage service for virtual machine. Block storage component provides an infrastructure management data volume and interacts with OpenStack computing service as well as offers data volume for examples.

F. Chen and Y. Luo (Eds.): Industrial IoT 2017, LNICST 202, pp. 72–82, 2017.
DOI: 10.1007/978-3-319-60753-5_8

Quantum network component provides networking service, which means to create IP address of virtual machine and manage the network system structure through API.

Keystone authentication component is responsible for access management and service catalogue. Amongst, access management takes charge of authorization and access establishment of users and involves concepts, including users, tenants and roles. Besides, service catalogue can only be provided for users after each service of OpenStack is registered in Keystone. Endpoint is the access point of the corresponding service.

Horizon component provides visualized GUI image interface and makes the users to operate various resources of OpenStack platform [5–8].

1.2 Command Line Management Tool of OpenStack

OpenStack cloud platform has two different methods to manage the cloud resources. The first one is to realize the management over cloud resources by Horizon, a GUI image interface based on Web while the second one is to manage the resources by portals of OpenStack command line.

(1) Check the OpenStack cloud platform service
nova-manage service list
(2) Create examples of virtual machine
nova boot [name of virtual machine] --flavor [type of virtual machine] --image [ID of virtual machine image] --security-groups default-nic --net-id = [ID of the network which the virtual machine belongs to]
(3) Stop, suspend and delete the virtual machine
nova stop [vm-name]
nova suspend [vm-name]
nova delete [vm-name]

2 System Design

Elastic resource allocation system is based on OpenStack cloud platform and realizes JAVA localized encapsulation of open API interface as well as allocates various resources in resource pool dynamically, which mainly include image management module, virtual machine management module, network management module and tenant information module, etc. in order to provide an excellent user interface.

2.1 System Software Structure

Elastic resource allocation system is divided into three layers. The bottom layer establishes elastic resource pool through OpenStack cloud platform and provides infrastructure service. The middle layer realizes the encapsulation of local components by using the open API programming interface of OpenStack and provides functional support for the upper layer. In addition, the upper layer realizes the management system of B/S structure based on J2EE technical design as well as manages and allocates various resources in resource pool. Figure 1 shows the system software structure.

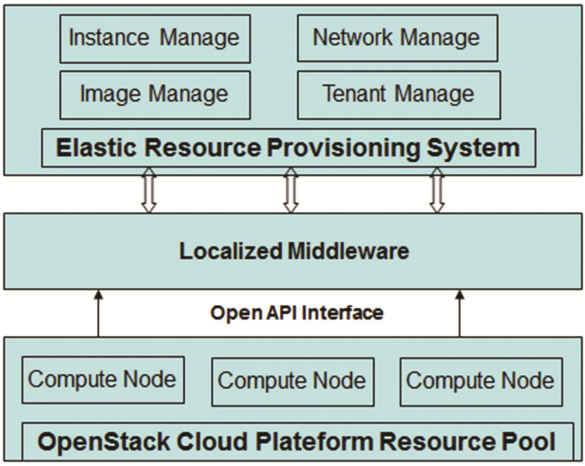

Fig. 1. Elastic resource allocation system software structure

2.2 Scheduling of OpenStack Cloud Platform Resource Pool

OpenStack cloud platform is established on the basis of server cluster. Then, the hardware resources at the bottom layer are abstracted as logic resources and are responsible for managing and scheduling virtual logic resources. The resource pool in the cluster includes the sum of resources of all the computing joints in the cluster. The logic resource pool develops API programming interface for the upper layer and provides transfer of infrastructure.

Resource scheduling refers to allocate the M heterogeneous and available resources to N mutually-independent application tasks in order to minimize the total task completion time and make full use of the resources. Resource allocation serves as a key component of cloud computing and its efficiency directly influences the working performance of cloud computing environment. The central controller of OpenStack, Nova is responsible for managing the resource computing of the whole cloud. Instead of providing any virtual capabilities, Nova makes the interactions between Libvirt API and host of virtual machine possible and provides external processing interface through Web service API. OpenStack makes use of the management program to provide the corresponding abstract relations between the application programs and the hardware. Therefore, a pool is equipped behind the server, network or storage device of each virtual machine, which makes the request response and resource allocation more flexible and effective [9].

Based on OpenStack open-source cloud platform framework, the functions of platform resource monitoring and dynamic resource scheduling are expanded into the computing module, Nova in order to include the monitoring and scheduling functions into this platform. Thus, corresponding function deployment can be accomplished when the platform is being deployed. In addition, service-oriented concept is adopted to manage the virtualized resources and realizes the optimal match between the physical resources at the bottom layer and the services at the upper layer by combining the

resource scheduling & allocation with service type and working load. Figure 2 shows the design framework.

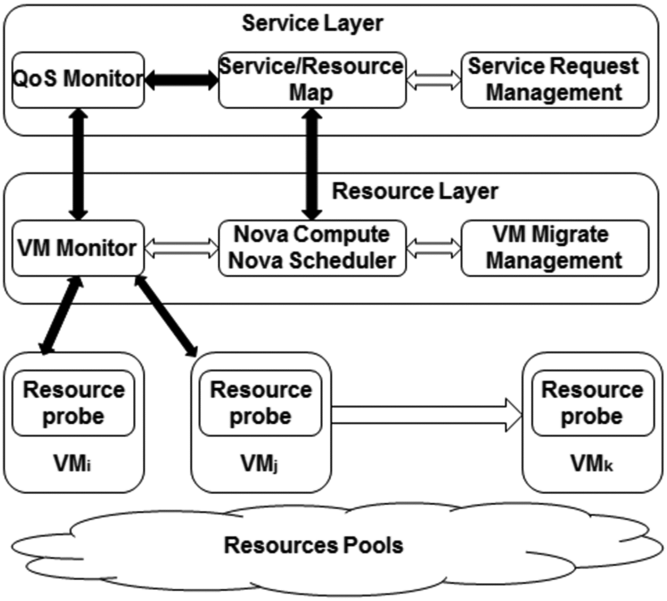

Fig. 2. Strategic framework of virtual resource scheduling

The scheduling algorithm locates all the virtual machines whose load is higher than the upper threshold in the virtual machine cluster and adopts the optimal adaptive algorithm to find the physical machine whose physical resource load is the highest but does not exceed the upper threshold for physical load after the scheduling process; then, the virtual machine is transferred to this physical machine and both the resource vector magnitude and the load of physical machine are updated.

2.3 Cluster Management Middleware

Multi-cluster management middleware is mainly responsible for shielding the multi-computing cluster distribution of the bottom cluster layer. The platform management layer transfers the middleware to accomplish resource integration and provides related information about each sub-cluster or sub-platform managed by the platform when operations, including virtual machine creation, storage management and information collection, need to be carried out in order to realize the goal of scheduling all the resources distributed across the whole platform during resource transfer.

Multi-cluster management middleware is a dynamic linkage module encapsulated through JNI technique by developing API interface provided by OpenStack cloud platform. On the basis of infrastructure platform, multi-cluster management, resource collection, storage service and virtual machine service are realized. In addition,

management middleware can not only transfer OpenStack platform resources but simplify and encapsulate the transfer process in order to provide realization of functional module for the application system at the upper layer and increase the flexibility and scalability of elastic resource allocation system. The realization process includes following steps:

(1) Receive the transfer request of functional module in elastic resource allocation system.
(2) According to the request type, write the corresponding processing script, which contains the resource scheduling command identified by OpenStack cloud platform, into the memory and implement it.
(3) Cloud platform cluster makes corresponding disposals according to the command and returns the results back to the middleware.
(4) Middleware converts the format of feedback data from cloud platform to send the data in a format which complies with system transmission protocol to the system module.

Table 1 shows the main contents encapsulated by JAVA localized middleware:

Table 1. Middleware interface encapsulation table

Name of encapsulation interface	Name of functions transferred by upper module	Middleware realizing function
Example of inquiring virtual machine	private native String novaListJNI()	JNIEXPORT jstring JNICALL novaListJNI (JNIEnv *, jobject)
Inquiry image	private native String novaImageListJNI ()	JNIEXPORT jstring novaImageListJNI (JNIEnv *, jobject)
Example of initiating virtual machine	private native String novaBootJNI (String flavorId, String imageId, String name)	JNIEXPORT jstring JNICALL novaBootJNI (JNIEnv *, jobject, jstring, jstring, jstring)
Example of stopping virtual machine	private native void novaSuspendJNI (String instanceId);	JNIEXPORT void JNICALL novaSuspendJNI (JNIEnv *, jobject, jstring)
Example of deleting virtual machine	private native void novaDeleteJNI (String instanceId);	JNIEXPORT novaDeleteJNI (JNIEnv *, jobject, jstring);

Middleware is responsible for encapsulating the scheduling commands and converting the format of data transmitted. The existence of middleware can greatly reduce the coupling among the layers in software structure and provide convenience for system function expansion in the future.

2.4 Elastic Resource Allocation System

Elastic resource allocation system realizes the management over various resources in cloud platform in modules and is mainly divided into five major modules: image management module, virtual machine management module, network management module, user information management module and auxiliary function module. Amongst, the image module achieves the functions of creating images by image files, viewing existing images and creating images by virtual machine snapshot; virtual machine module achieves the functions of creating, initiating, stopping and deleting virtual machine; user information module achieves the functions of importation, revision and view of user information; network module achieves the functions of generating floating IP and its binding with virtual machine; auxiliary function module achieves the functions of server performance inspection and real-time synchronization of system background data., etc.

The system background is realized by JAVA programming language and achieves control over major computing joints in cloud platform by class function, RmtShellExecutor through remote access.

2.4.1 Virtual Machine Management Module

Aiming at virtual machine resources, this module can realize the functions of virtual machine creation by images, management and control of virtual machine as well as view of virtual machine information.

(1) Creation of virtual machine. Select the type of image, set the configuration parameters (memory size, CPU and hard disk) of the virtual machine and generate virtual machines in batches.
(2) Management and control of virtual machine. Initiate, suspend, stop or delete designated virtual machine resources according to ID, name and type of virtual machine.
(3) View of virtual machine information. View the information about existing virtual machines in cloud platform, which mainly includes ID, name, status, network of virtual machines and operations supported by virtual machines.

2.4.2 Image Management Module

Images are master files used to derive virtual machines in cloud platform and create image files according to the standard image creation procedure for OpenStack cloud platform:

glance add name="win7" is_public=true container_format=ovf disk_format=raw < win7.img

Image management module can upload, create, view images and make images by virtual machine snapshot. Amongst, the snapshot function can use the storage of current state of virtual machine to copy virtual machines with the same contents. Figure 3 shows the snapshot function in image management module:

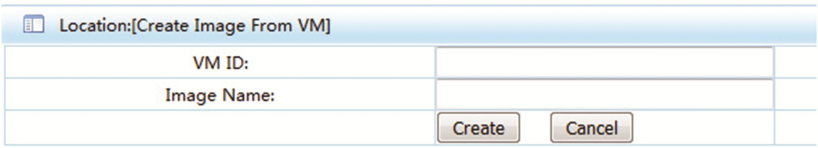

Fig. 3. Image management module snapshot function figure

2.4.3 Network Management Module

Each virtual machine operating in elastic resource allocation system corresponds to two IPs, one fixed inner network IP and one floating IP. The floating IP needs to be bond with virtual machine. Therefore, the network management module is responsible for batch creation and binding of floating IP.

2.4.4 User Information Management Module

User information management module is responsible for batch creation, deletion, revision and inquiry of user information in cloud platform. To facilitate the administrators to add batch user information, the administrators can fill in user information according to given format and put the information into an Excel sheet and then upload the sheet. The basic information about the virtual machine bond to the user can be inquired about in user information module.

2.4.5 Auxiliary Management Module

Auxiliary management module is responsible for viewing the operation state of server background resources, including the utilization rate of CPU, occupation rate of memory and utilization rate of exchange partition, etc. The auxiliary system administrators judge the operation condition of server in order to create virtual machine rationally and allocate corresponding hardware resources. Figure 4 shows the operation effect of auxiliary module.

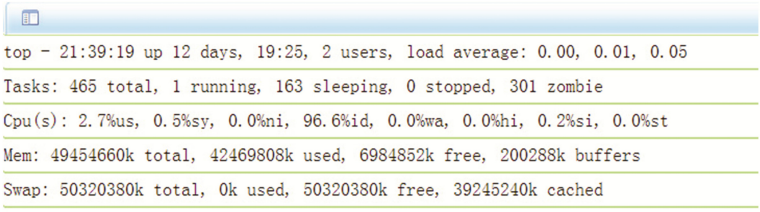

Fig. 4. Operation effect of auxiliary module

3 Key Technologies for System Realization

3.1 OpenStack API Authentication and Request Workflow

Authentication is required when the system accesses the OpenStack services. Firstly, authentication request should be sent to obtain the authentication token. Therefore, valid certificate must be provided in order to request the authentication token. When the system sends OpenStack API request, the token information is put into the X-Auth-Token head of HTTP request message. The token has a valid term and will become invalid after the term [10, 11]. OpenStack API authentication and request workflow process is as follows (Table 2):

(1) Access the authentication service access point of cloud platform, request for the authentication token. A valid certificate is included in the request sent, which contains following request parameter sheet:
When the request is successfully sent, the server will return back an authentication token.
(2) Place the token into the head of X-Auth-Token of HTTP request message to send API request. Keep using this token to send API request until the operation is finished or the server returns to 401Unauthorized [12, 13].
(3) When 401Unauthorized error occurs, please request for a new token.

Table 2. Request parameter of certificate table

Parameter	Type	Description
username (required)	xsd:string	Username. If you cannot provide username and password, token must be provided
password (required)	xsd:string	Password of this user
tenantName (elective)	xsd:string	Name of tenant. Both tenant ID and name are elective but cannot be used at the same time. If these two properties are designated, the service will return 400 error request
tenantId (elective)	capi:UUID	ID of tenant. Both tenant ID and name are elective but cannot be used at the same time. If these two properties are designated, the service will return 400 error request. If you do not know tenantId, you can send a “” as a request for tenantID and gain this ID from the returned message
token (elective)	capi:UUID	Token. If you cannot provide token, user name and password must be provided

3.2 Localized Interface Realization Technology

Localized interface is shortened as JNI (Java Native Interface) to provide API for realize communications between JAVA language and other languages. In this system, the virtual machine scheduling command in OpenStack cloud platform is encapsulated into executable scripts and then stored into memory. The background of elastic resource allocation system is realized by JAVA language, which adopts JNI technology to execute

the scripts in local servers and obtain relevant data to return back to system background. Take the virtual machine initiation function for example [14], the realization steps are as follows:

(1) Statically load the dynamic linkage library of localized interface encapsulation in system background function module. The code is as follows:

```
static
{
System.loadLibrary("novaList");
}
```

(2) Declare localized realization method, transfer the localized realization. The code is as follows:
private native String novaBootJNI (String flavorId, String imageId, String name);
(3) Accomplish localized realization in C language. The method is declared as follows:
JNIEXPORT jstring JNICALL Java_com_execute_ssh_ExecuteSSH_novaBootJNI (JNIEnv * env, jobject jo, jstring flavorId, jstring imageId, jstring name)

4 System Performance Test

During performance test, electromagnetic calculation algorithm, FDTD algorithm is operated on each virtual machine in elastic resource allocation system for performance test. This algorithm possesses the features of medium communication amount and large calculation amount and can carry out multi-core parallel calculation in a single machine or among multiple machines through network connection, thus achieving the test on calculation capability and network environmental calculation capability.

4.1 Calculation Performance Test for Single Virtual Machine

This test is carried out on the physical machine and the virtual machine created by elastic resource allocation system respectively. Table 3 shows the testing hardware configuration environment.

Table 3. Single-point test environment

Resource type	Resource name	Description	Quantity
Physical resource	DELL workstation	Octa-core CPU, 16 GB memory	1
Platform resource	Platform virtual machine	Octa-core VCPU, 16 GB memory	1

FDTD algorithm is used in these two environments for four calculations respectively. Table 4 shows the calculation results.

Table 4. Single-point test result

Resource type	First test results	Second test result	Third test result	Fourth test result	Average time
Physical resources	3997 s	4003 s	3978 s	3877 s	3963.75 s
Platform resource	4015 s	4211 s	4008 s	3997 s	4057.75 s

The average of the results of four calculations is obtained and thus the calculation time under platform virtual machine environment is about 1.02 times the calculation time of physical machine. The tests show that the operation performance in virtual machine environment is close to the performance of physical machine. Through platform automatic task operation and resource monitoring, the users can obtain the flexibility in operation and environment management with low performance loss.

4.2 Calculation Performance Test for Multiple Virtual Machine

This test adopts four sets of machines for calculation of FDTD cases. The testing environment for each set is the same as the environment in single machine test. Likewise, four calculations are carried out in these two environments respectively. Table 5 shows the calculation time results:

Table 5. Multi-point test result

Resource type	First test results	Second test result	Third test result	Fourth test result	Average time
Physical resource	4181 s	4103 s	4456 s	4480 s	4305 s
Platform resource	4691 s	4589 s	4983 s	5016 s	4819.75 s

Similarly, the average of the results of four calculations is obtained and thus the calculation time for virtual resources provided by the platform is about 1.12 times the calculation time of physical machine. The original calculation time is 89.3% of the calculation time under platform environment. Therefore, it can be seen that the performance loss of virtual machine environment provided by the platform is only about 10%. In addition, it can be seen that after the network delay is introduced, the calculation time of both the physical machine and the virtual machine environment increase. Although the time increases, this distributed calculation environment provides stronger calculation environment, which actually solves a more complex problem and obtains higher calculation precision. In the meanwhile, the calculation environment provided by the platform greatly simplifies the deployment and management of distributed calculation environment and facilitates the users.

5 Conclusion

Elastic resource allocation system realizes the scheduling and management of virtual resources based on OpenStack platform. Its main functions include virtual machine management and control, network resource management, user information management and image module management. The system provides humanized image interface and allocates the virtual resources flexibly and efficiently for the users.

References

1. Fei, X., Jing, Y., Liming, L.: Design and implementation of computer lab self-service platform based on openstack. Comput. Modern. **7**, 52 (2013)
2. Mingli, W., Tianhong, R., Yebai, L.: Application and research of resource management technology based on openstack private cloud platform. Ind. Technol. Innov. **2**(3), 334–341 (2015)
3. Shaoka, Z., Liyao, L., Xiao, L., Cong, X., Jiahai, Y.: Architecture and scheduling scheme design of tsinghua cloud platform based on openstack. J. Comput. Appl. **33**(12), 3335–3338, 3349 (2013)
4. Jinpeng, H., Qin, L., Chunming, H.: Research and design on hypervisor based virtual computing environment. J. Softw. **18**(8), 2016–2026 (2007)
5. Xianfeng, S., Junchuan, J., Xiaojun, Z.: Private cloud APCS platform design based on virtualization technology. Comput. Eng. **38**(8), 200–212 (2011)
6. Zhao, W.M., Wang, Z.L., Luo, Y.W.: Dynamic memory balancing for virtual machines. ACM SIGOPS Oper. Syst. Rev. **43**(3), 37–47 (2009)
7. Zhang, J., Gu, Z., Zheng, C.: Survey of research progress on cloud computing. Appl. Res. Comput. **27**(2), 429–433 (2010)
8. Wensheng, Z.: Application research of software virtualization in computer lab. China Electr. Power Educ. **8**, 113–114 (2012)
9. Mell, P., Grance, T.: The NIST definition of cloud computing (draft). NIST **800**(145), 7 (2011)
10. Zhuhua, W.: Analysis of Core Technologies of Cloud Computing. Posts & Telecom Press, Beijing (2011)
11. Qiang, X., Zhenjiang, W., Computing, C.: Application Development Practices. China Machine Press, Beijing (2012)
12. OpenStack. OpenStack Documentation [EB/OL]
13. http://docs.openstack.org. Accessed 16 Sep 2012
14. Peng, L.: Cloud Computing, 2nd edn. Publishing House of Electronics Industry, Beijing (2011)
15. Hua, Z.: Application of cloud computing technology in construction of university green computer labs. Value Eng. **11**, 180–181 (2012)