OpenStack-based Next-generation Cloud Resource Management

● Toshihide Yanagawa

FUJITSU Software ServerView Resource Orchestrator (ROR) is a middleware product for more efficiently operating and managing private clouds. In recent years, many companies have moved to aggregate their internal hardware resources, creating private clouds, while actively utilizing public clouds to reduce system deployment costs. These have produced a greater need for flexible operations of private clouds, demand for the ability to use the diverse application programming interfaces (APIs) that are available on public clouds on private clouds as well, and demand for the ability to centrally manage both public and private clouds. Fujitsu offers ROR to respond to this need. This paper introduces ROR, which uses OpenStack open source software to centrally and easily manage the resources of both public and private clouds.

1. Introduction

Utilization of cloud service is growing toward optimization of the way companies use information and communications technology (ICT). FUJITSU Software ServerView Resource Orchestrator (ROR)[1] has provided operation and management functions for private clouds, but demand is steadily rising for integration with public clouds. Open source software (OSS) has also made tremendous strides in cloud management. Fujitsu is planning to respond by providing cloud management functionality in ROR by using OpenStack,[2] which has gained prominence as a typical model OSS cloud software, as well as constantly enhancing the software.

This paper introduces past ROR initiatives, the functions and features of OpenStack, the synergistic benefits of using ROR with OpenStack, the ROR functions which provide added value to OpenStack, and future challenges with regards to using OpenStack in ROR.

2. Past initiatives

To use internal corporate ICT resources effectively, ROR has provided the following functions in addition to hardware resource management functions.[3]

1) Virtualization
- Virtualizes resources such as central processing units (CPUs) and memories in order to aggregate servers
- Provides greater flexibility in management and response to hardware failures through physical server and input/output (I/O) device virtualization
- Reduces workload of design through networking device virtualization

2) Standardization
- Handling systems composed of multiple servers and network segments as logical platforms, and using a template approach, makes it possible to deploy uniformly configured arrangements on a system-by-system basis.
- Integration with various hypervisors (VMware ESXi, Hyper-V, Xen, KVM, Solaris Zones, Oracle VM) and establishment of definitions in advance makes uniform structure and operation possible even in multi-VDI (Virtual Desktop Infrastructure) environments.
- Separation of roles, such as physical resource (infrastructure) administrators in charge of servers, storage, and networks, virtualized resource administrators, and resource users, with separately defined resource access and operation

permissions

3) Automation

- Usage of self-service portals and automation of application and operation reduces management workload.
- Automation of server, storage, and switch configurations makes it possible to change server roles.

By closely tying these functions to Fujitsu's hardware platforms, composed of servers, storage, networks, and the like, Fujitsu has been able to provide systems with greater degrees of availability and reliability. Fujitsu has also provided a unified view which smoothes over the functional variations between the previously mentioned hypervisors, resulting in improved operability.

3. OpenStack's functions

This section will focus on the OpenStack functions which are integrated with ROR.

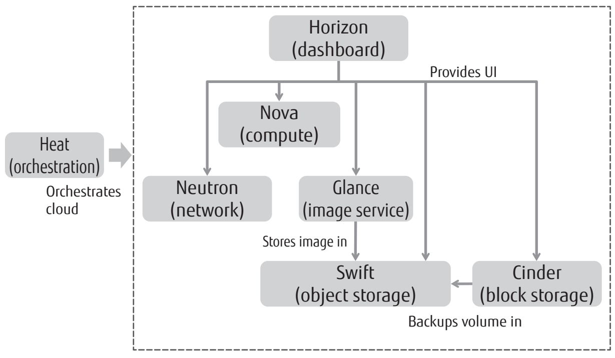
OpenStack is composed of multiple services (which will be called "components" in this paper in order to clearly indicate that they are software structural units).

These components are interrelated, as shown in **Figure 1**. They are loosely connected using application programming interfaces (APIs) in order to make them more independent.

Below are the project names and roles of the major OpenStack components.

1) Nova

Virtual machine lifecycle management. Nova generates, schedules, and deletes virtual machines on demand.



Source: http://docs.openstack.org/ja/trunk/install-guide/install/apt-debian/content/ch_overview.html

Figure 1
OpenStack component configuration.

2) Glance

Virtual machine disk image management. Glance handles guest OS provisioning.

3) Neutron

Provider of network configuration definition services. Neutron provides functions that enable users to easily build virtual environment networks via API.

4) Swift

Provider of object storage services. Swift provides API for easily replicating data and scaling out.

5) Cinder

Provider of block storage services. Cinder makes it possible to use specific functions provided by physical storage devices by creating plug-in functions for them.

6) Horizon

Provider of dashboard functions. Horizon provides a Web management screen, from which users can check resource usage status, change virtual machine configurations, etc.

7) Heat

Provider of automation mechanisms. Heat uses the APIs of various OpenStack components as defined in advance in templates, automating the processes of configuration creation and definition configuration.

Another included component is Ironic, an incubation project for managing physical servers. There are plans to integrate the Ironic component in ROR as well.

Using OpenStack makes it possible to create virtual machines, operate storage and network resources, and carry out operations such as authentication and permission on a highly detailed basis using the APIs supplied by OpenStack's components. This can flexibly create cloud environments.

4. Synergistic benefits of integration

Not only does embedding OpenStack in ROR make it possible to use OpenStack's functions as they are, but it also provides the following three cloud operation benefits.

1) Operation flow clarification through highly detailed role division

In ROR, roles and permissions are defined and standardized by user type, such as infrastructure administrator, tenant administrator, and tenant user. OpenStack has a high degree of role definition freedom, often leading to confusion in actual operation. The organized roles of ROR can clearly define operation flow.

2) Improved management through the introduction of the concept of "logical platforms"

ROR defines business systems composed of multiple servers as logical platforms. It uses the logical platform as management units of the system, enabling users to visualize and carry out batch operations and improving management.

3) Improved availability and reliability through detailed hardware control

ROR is closely integrated with hardware such as FUJITSU Server PRIMERGY and FUJITSU Storage ETERNUS, providing functions for improving availability and reliability. For example, it can monitor the signs of impending server hardware failures, migrating virtual machines to healthy servers before the failures occur, and preventing hardware problems from interrupting system operation.

In addition to using OpenStack's versatile APIs in highly flexible cloud management, ROR's standardization structures can also be introduced to make it easy to implement cloud management. Adding ROR's highly detailed control capabilities to OpenStack's unified hardware control capabilities will make versatile operation management possible while also providing business systems with high levels of availability and reliability.

5. ROR's added value

OpenStack has developed rapidly, led by an active community, but the focus of this development has been on new functions, and further development is expected in the area of ease of operation to make it usable for business purposes. ROR provides value added availability and usability functions, essential for business usage, based on its extensive track record.

For improving availability, ROR provides the following availability improvement functions, in addition to the aforementioned failure prediction monitoring.

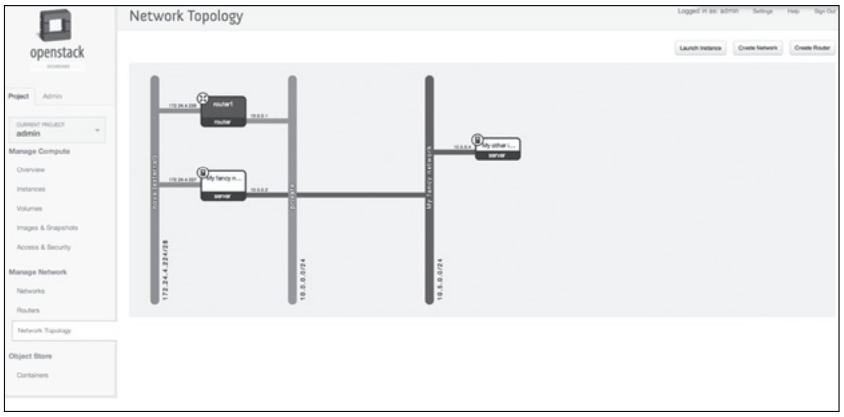
1) Server repurposing

Servers can be repurposed by switching to another physical server's system disk. For example, if a development server is defined as a spare business operation server, in the event of a failure to the main business operation server, the development server can be used as a business operation server in its stead.

2) Disaster recovery

In the event of a disaster affecting an operation site, the configurations of the business systems located at the disaster-affected site can be replicated in a disaster recovery site, allowing business operations to continue.

ROR also provides a conventional network viewer function, simplifying operation and improving usability. **Figure 2**[4] shows how networking device configuration is shown in OpenStack's dashboard function (Horizon). ROR's network viewer, which is an existing function, can display the relationships between virtual machines, virtual switches, physical servers, and physical switches in a visual, easy-to-understand manner, down to the individual network port level, as shown in **Figure 3**. There are also plans to outfit the network viewer with the ability to make connectivity configuration changes simply by modifying the on-screen configuration diagram.



https://www.openstack.org/assets/software/grizzly/6NetworkTopologyView2.png

Figure 2
OpenStack dashboard (network topology) screenshot.

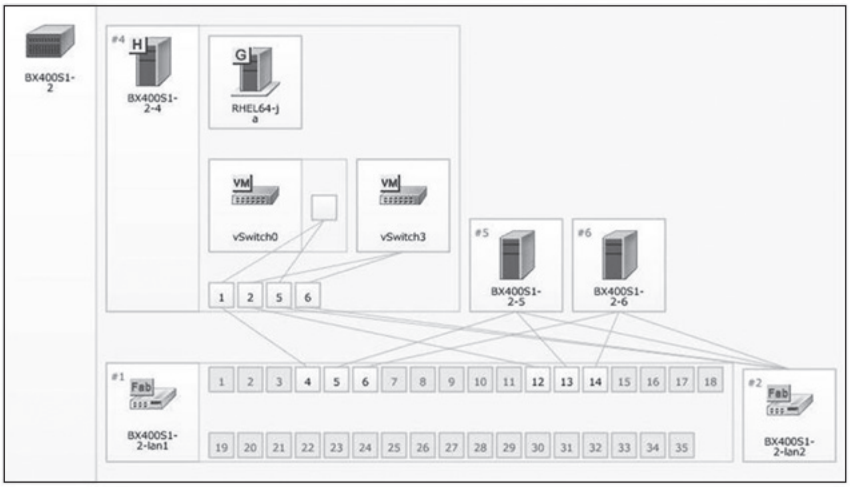


Figure 3
ROR network viewer screenshot.

6. Future challenges

ROR is a cloud management middleware product for high availability, high reliability systems. The usage of OpenStack, open source software, in this type of middleware presents the following challenges.

The OpenStack community has been very active, and major enhancements are expected in the future. This may result in a lack of function continuity (compatibility) between current and past versions. A key challenge, then, will be to continually provide interfaces that ensure compatibility.

Customer operation may require the use of revised or additional OpenStack functions. There may be cases where these functions must be provided to customers before they can be developed by the OSS community. In this case, Fujitsu developers quickly catch up with the frequent program changes made by the active OSS community and make appropriate responses to customers while providing feedback to the OSS community.

In addition to this ongoing support, we are also working to ensure we can continuously provide added value through ROR to the commodity cloud management infrastructure of OpenStack.

7. Conclusion

This paper described FUJITSU Software ServerView Resource Orchestrator, Fujitsu's public and private cloud operation infrastructure middleware. Using OSS for cloud operation management makes it possible to speedily handle global standard operation management infrastructure while providing Fujitsu's highly detailed server, storage, and network hardware control capabilities.

We plan to keep up with the rapid progress of OSS, maintain operation compatibility, and deliver product enhancements, led by platform compatibility and scalability, creating the future Fujitsu next-generation cloud resource management.

References

1) Fujitsu: FUJITSU Software ServerView Resource Orchestrator.
http://www.fujitsu.com/global/products/software/infrastructure-software/resource-management/serverview-resource-orchestrator/
2) The OpenStack Project.
http://www.openstack.org/
3) H. Matsumoto, et al.: Dynamic Resource Management in Cloud Environment. *Fujitsu Sci. Tech. J.,* Vol. 47, No. 3, pp. 270–276 (2011).
http://www.fujitsu.com/downloads/MAG/vol47-3/paper04.pdf
4) The OpenStack Project - OpenStack Grizzly.
https://www.openstack.org/software/grizzly/

Toshihide Yanagawa
Fujitsu Ltd.
Mr. Yanagawa is currently engaged in development of cloud management software.