A Platform for Tenant Network Provisioning with Provisioning Template

Yoji Ozawa, Yoshiko Yasuda, Yosuke Himura
Yokohama Research Laboratory, Hitachi, Ltd., Kanagawa, Japan

***Abstract*—Quick provisioning of tenants in multi-tenancy data centers is one of major challenges. Provisioning is time-consuming and error-prone due to the need to configure network devices with hundreds of parameter values (e.g., VLAN ID, IP address) determined according to operational rules. Past works have developed systems to automate such provisioning operation, but a crucial problem stems from the high program development cost to adopt multiple data centers.**

In this paper, to solve this problem, we define the provisioning processing as "provisioning template" described in editable external files. The key components of the provisioning template are parameter value decision rules (e.g., operational rules of parameter assignment and abstracted representation of parameter dependency) and a mapping data between tenant and infrastructure. Since the decision rules are not designed for particular configuration items, we can provision a tenant with various configuration items. Our provisioning platform prototype shows the platform can adopt additional data centers with less development cost than a conventional system.

I. Introduction

The use of a multi-tenancy data center is one of the recent fashion in the IT business environment. Here, "tenant" is defined as a logical infrastructure for a business system, and "multi-tenancy data center" as data center which hosts multiple tenants on a single physical network.

Provisioning a tenant is time-consuming and error-prone [1], resulting in the needs for automation of provisioning operation. We here note that "provisioning" means the operation to create an instance of tenant network over an infrastructure, following the steps of deciding parameter values for network devices, selecting appropriate devices to deploy the tenant, making configuration commands with those parameter values, and inputting these configuration commands into network devices. Conventional multi-tenancy data centers are composed of various network devices such as switches and appliances like firewalls and are managed according to various operational rules to decide parameter values to be configured. The operational rule is not only to assign arbitrary values from resource pools but to decide values with operator-specific rules [2]. For example, the rule assigns consecutive numbers to the third octet of an IP address and decides VLAN ID's single digit according to a subnet position.

There have been several systems which automate such provisioning operation [3] [4]. These systems manage network composition with a database, obtain and calculate parameter values by querying the database and generate configuration commands by substituting the values into configuration templates as shown in Figure 1(a). The configuration template has been defined as a set of configuration commands, in which parameter areas are blank.

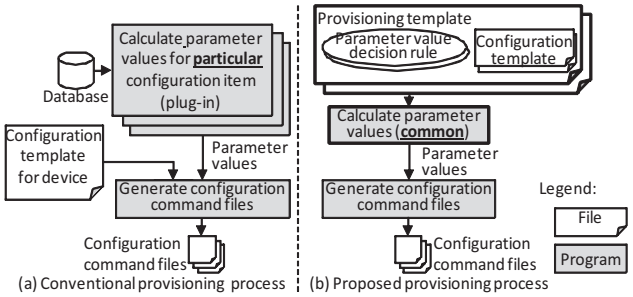

Fig. 1. Conventional and proposed provisioning processing

However, a crucial problem is the incapability of directly applying those systems to evolving data center environments. It is difficult to apply the system to additional data centers with different physical infrastructure and a different tenant composition. In order to deal with these differences, we will spend much time and financial cost for a program development.

In this paper, to overcome this problem, we propose a provisioning platform with a provisioning template, which is an easily-editable external file. The provisioning template includes the configuration template as well as "parameter value decision rule" defining a parameter calculation processing, which has been a hard-coded program in conventional systems as shown in Figure 1(b). We define types of the decision methods and input data to decide parameter values (i.e., assigned pool ID and referring parameter ID, etc.) in the decision rule. The provisioning platform calculates parameter values according to the decision rules. Since these decision rules do not target particular configuration items, we can provision a tenant with various configuration items by just modifying the provisioning template files. Consequently, our platform can be easily applied to additional data centers.

II. Requirements for Tenant Provisioning in Multi-Tenancy Datacenter

A. Tenant Provisioning with Various Configuration Items

Each data center has different types of configuration items for tenant network. For example, the configuration items are a VLAN, an IP address on VLAN interface (IF), a firewall's virtual router, etc. Therefore a provisioning processing needs to handle various configuration items flexibly.

B. Tenant Provisioning on Various Physical Infrastructure

It is necessary to construct a tenant on various physical infrastructures while preserving a tenant composition. For

978-3-901882-50-0

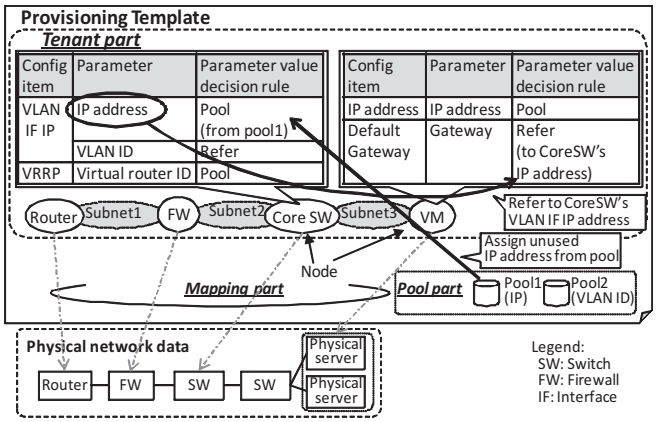

Fig. 2. Example of provisioning template structure and parameter calculation

example, we need to build tenants on both active data center and data center for disaster recovery (DR). Active data center's physical infrastructure is usually different from the DR data center's one.

C. Wide Coverage of Each Provisioning Template

It is required to apply a single provisioning template to as many as possible of tenants, because the structures of tenants can frequently be different. For example, tenants with 3-tier system composition are the same in the viewpoint of network composition, but the number of virtual machines (VM) can be different. If we make provisioning templates for every number of VMs, the number of provisioning templates becomes massive, which is inefficient.

III. Provisioning Template

We propose a provisioning template, which defines the provisioning processing. We design the template structure to satisfy the requirements discussed in section II.

A. Provisioning Template Structure

We define some data to automate a tenant provisioning in the template and the key components of the template are the parameter value decision rule and a mapping data between tenant and infrastructure. Figure 2 shows an example of a provisioning structure. The template consists of a tenant part, a mapping part and a pool part. Physical network data are generated from actual network device data to obtain current physical network.

The tenant part models the composition of a tenant and has configuration items to provision a tenant. The tenant part consists of nodes and subnets.

- Nodes represent logical devices (e.g., switch, router, firewall and VM) composing tenant. Each node includes configuration items (e.g., "VLAN IF IP" for a core switch), parameters for the configuration items (e.g., IP address) and decision rules to decide a parameter value (e.g., the IP address is assigned from the pool1). Those components are described in an abstracted manner.
- Subnets define IP-level structure. A subnet includes decision rules for VLAN ID and the network address of a subnet.

To satisfy the requirement A, we define the decision rule as the rule which does not target particular configuration items but is generalized for a network parameter as detailed in Section III-B1. To satisfy the requirement C, we introduce a "multiplicity" for the node and the configuration item as detailed in Section III-B3.

The mapping part has relation data between tenant nodes and physical devices. A generated configuration command is applied to a corresponding physical device. The tenant part is defined separately from the mapping part. We modify only the mapping part and then we can provision a tenant with the same tenant composition on different physical infrastructures as mentioned in the requirement B. We explain the detail in section III-B2.

The pool part has a pool data to decide a parameter value. There are two pool types: "ID pool" and "IP address pool". We define ranges of value for ID pool and network addresses and default mask length, which is used in assigning network address from IP address pool. A provisioning platform assigns unused ID and IP address referring pool instances to decide parameter values in provisioning.

B. Characteristics of Provisioning Template

1) Generalized Parameter Value Decision Rule : It is required to provision various configuration items for a tenant network. Hence we define the decision rule as a generalized rule for network parameters. According to actual operations, we found that most of parameter values can be determined by the following two decision methods and calculation of values decided by these decision methods.

- Assigning from pool: We decide a value by assigning unused ID or IP address from pool. It is therefore necessary to manage resources as a pool and usage status.
- Refer to other parameter value: Since network parameters are related to each other, we usually decide a value by referring to other parameter values.

Therefore, we define two parameter value decision rules "assigning from pool (*pool*)" and "refer to other parameter value (*refer*)." We define the pool for the rule "*pool*" and the referred parameter for the rule "*refer*." The rule "*refer*" can refer to the subnet data's VLAN IDs and network addresses. We also define an arithmetic and string calculation of values decided by these rules. A provisioning platform decides parameter values automatically from just these decision rules in provisioning.

For example, as illustrated in Figure 2, the decision rule of the core switch's parameter "IP address" is "*pool* (from pool1)", and hence the parameter value is decided by selecting unused IP address from IP address pool. Because the value of parameter "Gateway" of VM is usually IP address of core switch, a decision rule is "*refer* (to core switch's IP address)." A provisioning platform decides the parameter values by assigning from pool or referring to other parameter value.

2) Mapping Data between Tenant Node and Physical Device : The tenant part is defined separately from the mapping data indicating configuration target physical device

Fig. 3. Example of the provisioning template

to provision a tenant on various physical infrastructures. We modify only mapping data and then we can provision a tenant with the same tenant composition on different physical infrastructures.

We map the node "VM" to a group of physical servers. The platform can assign "VM" to one of physical servers automatically or manually (according to an operator's demand).

3) Multiplicity for Node and Configuration Item : We introduce a "multiplicity" for the node and the configuration item for a provisioning template to deal with tenants which has the different number of nodes. We use the multiplicity to define the number of the nodes and the configuration items as an undecided number. When an operator provisions a tenant, an operator decides the multiplicity value. It is possible to provision some tenants with a single provisioning template.

Moreover not only an operator decides the multiplicity value but also a provisioning platform decides the multiplicity value automatically. For example, if the number of VMs is changed, the number of configuration items "firewall policy (access control list)" which destination is the VM has to be changed as well. A provision platform generates configuration items "firewall policy" as many as the number of VMs.

C. Provisioning Template Format

An example of the provisioning template is shown in Figure 3. The provisioning template is an XML document to easily define a document structure with XML Schema, etc.

We define the tenant part and the mapping part in a single file. On the other hand, we define the pool part in another file, because the pool part is shared with multiple the tenant parts. The file including the tenant part and the mapping part has *tenantTemplate* element corresponding to a single tenant as a root element. This element has node data and subnet data. *Node* element has a parameter data and configuration item data. The decision rule is defined in *parameter* element. For the decision rule, a decision rule type, an used pool and a referred parameter are defined as a *decisionType* element, a *poolId* element and a *referId* element respectively. *ConfigItem* element has *usingParameters* elements for parameters which

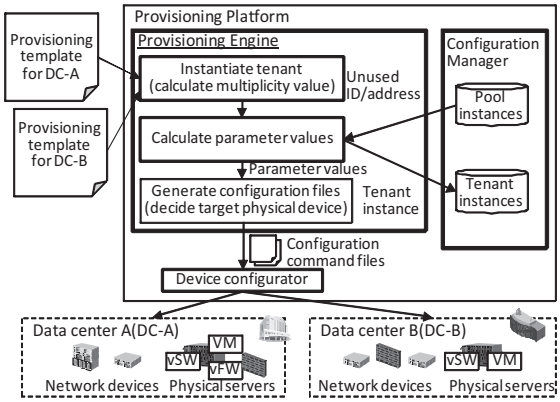

Fig. 4. Provisioning platform architecture and provisioning processing flow

are used by the configuration item. The mapping part is defined with *mapping* element. *Mapping* element has a node and a physical device which is a configuration target device. The pool part is defined with *pool* element which has pool type and a range of resources.

IV. Tenant Provisioning Platform

A. Architecture

Figure 4 shows our provisioning platform architecture. The provisioning platform consists of the provisioning engine executing provisioning processing, the device configurator inputting generated configuration commands into devices, and the configuration manager. The provisioning template files for each data center are input into the provisioning platform.

The provisioning engine generates configuration commands with the provisioning template. The only thing we need to do is the modification of the mapping data in order to configure the virtual firewall instead of the physical firewall. The configuration manager manages configuration data, which includes tenant instances and a pool data which manages the usage status for resources (e.g., IDs and IP addresses). "Tenant instance" is data of a provisioned tenant (i.e., calculated parameter values and generated configurations).

B. Provisioning Processing

The provisioning processing is proceeded along with the following three steps:

1) Instantiating Tenant : The provisioning engine instantiates a tenant. At this time, the provisioning engine calculates the multiplicity values of nodes and that of configuration items based on an operator's request and parameter dependencies, and generates instances as many as the multiplicity value.

2) Calculating Parameter Values : The provisioning engine calculates all parameter values in accordance with the decision rules. At first, the provisioning engine decides parameter values of a decision rule "*pool*" by assigning unused ID or address from a pool instance in the configuration manager. After that it creates a directed graph expressing reference relationships using the parameters which have a decision rule "*refer*" and referred parameters. A vertex of the directed graph is the parameter and an edge is a reference relationship between parameters defined in the decision rule. An edge

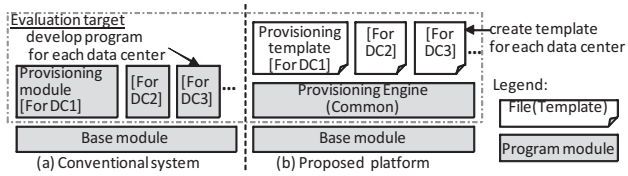

Fig. 5. Target for evaluation of development cost

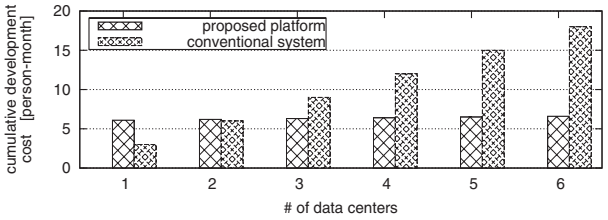

Fig. 6. Comparison of cumulative development cost

has the direction from a referring parameter to a referred parameter. It executes the depth-first search and calculates a parameter value with defined arithmetic and string calculation if values of all children vertices are decided.

3) Generating Configuration Commands : The provisioning engine generates configuration commands by substituting calculated parameter values into configuration templates and selects physical devices which are to be configured according to the mapping data. Then the device configurator applies generated configuration commands to the selected devices.

V. Evaluation

We evaluate the superiority of our platform regarding a development cost to handle multiple data centers. We compare cumulative cost between a conventional system and the proposed platform.

We estimate the development cost of a conventional system and the proposed platform to apply them to additional data centers. Figure 5 shows program modules and templates to be evaluated. In the conventional system, we need to develop a provisioning module for each data center in order to handle different decision rules and configuration commands. Therefore we estimate a program development cost of the provisioning module. On the other hand, in the proposed platform, we estimate a program development cost of the provisioning engine and a creation cost of the provisioning template. Since we can use the provisioning engine for multiple data centers, we develop it once. Moreover a modification of a configuration templates included in the provisioning template can treat difference of physical devices. We only create the provisioning template after we apply the platform to a first data center.

To evaluate the efficiency of our provisioning platform, we implemented both the provisioning module of the conventional system and the provisioning engine of the proposed platform.

The codes of the provisioning module were approximately 3,000 steps equivalent to three person-month, while the codes of the proposed platform were about 6,000 steps equivalent to six person-month. In addition, the provisioning platform needs additional cost to create the provisioning template. It was approximately three days which are equivalent to a 0.1 person-month. Figure 6 shows the comparison of cumulative development costs between the conventional system and the proposed platform. The horizontal axis is the number of data centers which are applied to the system.

The development cost of the proposed platform is about six person-month. Since we only create the provisioning template to apply to an additional data center and the cost is relatively small, the cumulative development cost increases slightly. On the other hand, we spend much cost for the development of the provisioning module for additional data centers. Therefore the cumulative development cost increases sharply. By considering the number of the data centers, the cost of the conventional system is smaller than that of the proposed platform within two data centers. However, the cost of the proposed platform is smaller at more than three data centers and the gap of the cost becomes large according to the number of the data centers.

VI. Related Work

Several works related to configuration template-based network configuration management and generation have been conducted [5] [6]. Enck et al. [5] propose the configuration generation and management with a database and a configuration template *configlet*. However, their system is for a carrier network. They do not consider that every tenant has different number of nodes like VM, and hence their system does not necessarily have enough capability of tenant management in data centers. On the other hand, our platform can deal with some tenants having different number of nodes with a single template. It is therefore suitable for multi-tenancy data centers.

VII. Conclusion

We have developed the provisioning platform which automates provisioning operations. Instead of hard-coded programs for a provisioning processing, our platform provisions with a provisioning template described in editable external files. The key components of the provisioning template are decision rules and mapping data between tenants and infrastructure networks. Since these decision rules are not designed only for particular configuration items, we can use these decision rules to provision various configuration items. We showed, by implementing the proposed platform, that the platform can handle additional data centers with less development cost. In our future works, we further intend to evaluate a capability for variety of tenant compositions.

References

[1] D. Oppenheimer, A. Ganapathi and D. A. Patterson, "Why do internet services fail, and what can be done about it?," in *USENIX USITS'03*, p.15, 2003

[2] "Cisco Virtualized Multi-Tenant Data Center, Version 2.1, Implementation Guide," http://www.cisco.com/en/US/docs/solutions/Enterprise/Data_Center/VMDC/2.1/implementation_guide/vmdcImpl21.pdf, 2011

[3] D. Caldwell, S. Lee and Y. Mandelbaum, "Adaptive parsing of router configuration languages ," in *IEEE INM'08*, pp.1–6, 2008

[4] B. Vanbrabant and W. Joosen, "Integrated management of network and security devices in IT infrastructures," in *CNSM'11*, pp.375–379, 2011

[5] W. Enck, P. McDaniel, S. Sen, P. Sebos, S. Spoerel, A. Greenberg, S. Rao and W. Aiello, "Configuration management at massive scale: system design and experience," in *USENIX ATC'07*, pp.73–86, 2007

[6] J. Gottlieb, A. Greenberg, J. Rexford and J. Wang, "Automated provisioning of BGP customers," in *IEEE Network*, Vol.17, Issue 6, pp.44–55, 2003