Cloud-based SI for Improving the Efficiency of SI in the Cloud Computing by Means of Model-Based Sizing and Configuration Management

SHIMAMURA Hisashi, YANOO Kazuo, KAJIKI Yoshihiro, KURODA Takayuki, NAKANOYA Manabu

Abstract

With the arrival of the age of cloud computing, the system integration platforms are changing from the servers and storages of the on-premise age to become virtualized cloud platforms. This paper introduces the SI technologies of the cloud age, which is improving efficiencies in estimation, development, sizing, testing, and rollout of production environment by applying automation technologies that are taking advantage of the features of the cloud system. We focus on the overall picture, including the model-based design support technology (CARDO), configuration management technology (Alchemy) and on template based provisioning.

Keywords

SI process, sizing, configuration management, auto-construction, customizing, template

1. Introduction

The progress in server/network virtualization and cloud technologies has been accompanied by the dissemination of system integrations (SIs) that utilize cloud platforms, as well as by the hardware installed in the traditional on-premise environments. The cloud technology accelerates the provisioning of the infrastructural layer such as virtual machines (VMs) and virtual networks. However, the processes that feature high dependence on individual operators can cause delays in the building of the overall system, including the middleware and application layers in the upper stages. The non-functional design that requires high skill for accurate estimation and the setting/deployment process that requires knowledge of middleware and applications are highly dependent on individual skills and therefore tend to cause delays and may require redesigns. These processes have the potential of becoming a factor in obstructing agility which is one of the features of cloud platforms.

In this paper, we describe the concept of cloud based SIs that can improve the efficiency and agility of these processes as well as the key technologies that support the cloud based SI. Additionally, we also discuss a cloud-based SI support tool that was prototyped in FY2014.

2. Cloud-based SI

2.1 The Cloud-based SI Concept

The objective of the cloud-based SI is to improve efficiency and to implement automation of the non-functional design process that features a highly individual skill dependency. The building/deployment process of the testing and production environments based on the configuration information of the building target system is also supported. **Fig. 1** shows an image of the cloud-based SI. The configuration information is described in a highly reusable format so that it can be adapted with minimum labor even in the case of a product recombination or a server configuration change and according to the customer requirements. Below, we describe the concept of efficiency improvement and the automation of each process.

(1) Sizing

Within the cloud environment, the use of the auto scaling function can reduce the need for a rigorous performance design, but this does not mean that it makes the performance design unnecessary. This is because consideration of the hard-to-scale middleware such as the RDBMS (Relational DataBase Management System) and shared re-

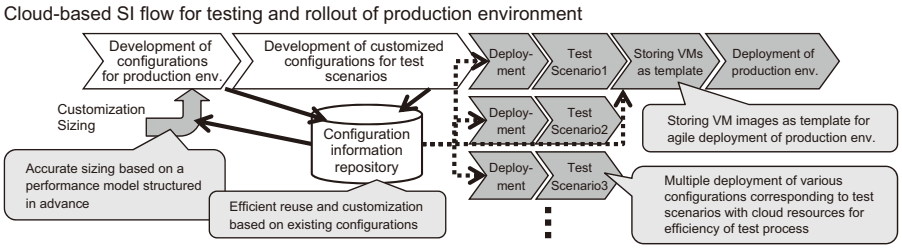

Fig. 1 Cloud-based SI flow for testing and rollout of production environment.

sources such as networks and storages will continue to be required. Furthermore, whether the project is to be started or not cannot be decided unless the total cost is calculated by estimating the necessary resources.

However, when a platform is shared by multiple tenants, which is one of the features of the cloud environment, the effects of the system loads of the other tenants is hard to predict, so precise sizing is generally difficult to achieve. We therefore adopt the policy of performing "rough" sizing in order to estimate the existence of bottlenecks, the load that can be withstood if any bottleneck exists, and the total cost. Although sizing is referred to as being "rough," the need for quantitative evaluation with a certain accuracy is presupposed and any large deviation from the specified non-functional requirements is not acceptable.

(2) Building/deployment of test environments

With regard to integration testing before system release, it is necessary to build a system execution environment equivalent to that of the actual production to try configuration changes such as variations in settings, according to various test scenario requirements and insertions of mockup modules. The on-premise type SI completes test scenarios sequentially by performing configuration changes and failbacks. Meanwhile, the cloud-based SI designs a system configuration assuming the use of deployment/setting tools with auto building capabilities such as Chef[1] and Puppet[2]. It thereby creates the new system execution environment for each test scenario based on the system configuration every time that one is required. The settings and configurations modified by each test are not applied to the execution environment but to the configuration definition, and the execution environment is scrapped after testing without failback. This technique makes it possible to build execution environments concurrently for several test scenarios, based on the rich resources of the cloud environment and to therefore shorten the time spent on testing.

(3) Building/deployment of the production environment

Traditionally, the system execution environment that has passed testing is used as the production environment of the on-premise type SI. With the cloud-based SI, the system execution environment that has passed testing is saved as a golden image. At this time, it is not only the single VM image that is set in the system but all of the VM images, network configurations and associated resources are saved as cloud templates. This technique is used as the basis of building the system execution environment when the production environment is released. Thereby enabled are an increase in the speeds of application of the entire system configuration to other similar systems, the creation of a reproduced environment in the case of a fault and the building of an alternative environment in the case of a natural disaster.

2.2 Key Technologies Supporting Cloud-based SIs

(1) Model-based design support technology "CARDO"

The model-based design support technology CARDO enables semi-automation of non-functional designing by advancing the traditional non-functional design support technology based on non-functional evaluations on a model.

This technology improves the sizing efficiency achieved by use of the cloud-based SI. The main features of this technology are listed below.

- **Non-functional evaluation using system model**
  CARDO evaluates the performance and availability of a system based on a system model and given design parameters. More specifically, it performs simulation or analytical calculation after converting the model into a hierarchical queuing model for the performance evaluation, or into a fault tree or stochastic reward net model for the availability evaluation. Since the hierarchical queuing model and the stochastic reward net model do not require detailed knowledge, they can be defined even by non-specialists in performance engineering or availability evaluation.
- **Fast search of optimum design parameters**
  By repeating non-functional evaluations CARDO searches the optimum design parameter combinations that can meet the given non-functional requirements. This is a kind of combination of optimization problems and it has succeeded in increasing the search speed

greatly by introducing optimum heuristics and utilizing the approximation solution obtained by analytical calculation as a relaxation problem.

(2) The system configuration management technology "Alchemy"

The system configuration management technology "Alchemy" offers a mechanism that inputs a system configuration combining components in a repository and the procedure for building the target system is thereby derived. The efficiency of system execution environment building via configurations that vary between cloud-based SI scenarios may thus be improved. The features and technological advantages of this technology are listed below.

- **Improved efficiency by reusing accumulated knowledge**
  The build script including middleware products, application installation method and setting the items used with the system are componentized in a framework unique to Alchemy and these are stored in the repository. Componentization using this framework enables the description of various system configurations as sets of these components, thereby enhancing the reusability of previously described build scripts. The system configurations described in this way can also be stored as optimum practices in the repository so that reproduction and application to other similar systems are facilitated.
- **Easy customization achieved by separating the provisioning requirements from the logical configurations**
  The configuration model of Alchemy describes the entire system as a set of components by following the concept of the Service Component Architecture (SCA).3) Thus, version upgrading of components, interchange with other products and modification of the setting parameters per component are facilitated. In addition, it separates the provisioning configuration that states the specifications and the number of servers allocated to building the target system from the logical configuration described above. The system makes it possible to recompile the provisioning configuration independently so that the server configuration may be changed easily according to requirements for each building. This easiness of customization reduces the amount of labor required to change settings and configurations every time for each test scenario.
- **Auto deviation of deployment procedure without inconsistency**
  When a user of Alchemy describes a system configuration by combining the configuration components in the repository, the dependences between components are defined. For example, a dependence is found in the fact that it is necessary to install the Java language execution environment before installing the Java EE server. Alchemy derives the building procedure of the entire system by solving the issue of dependences and instructs execution of the build scripts required for the servers specified in the provisioning configuration in an order that does not provoke inconsistency.

(3) Template-based provisioning

The template-based provisioning is a function of NEC Cloud IaaS (NECCI) that performs simultaneous provisioning of multiple VMs, networks and storages composing the targeted building system. It improves the efficiency of the production environment building of the cloud SI system that has passed the testing.

- **Simultaneous provisioning of multiple resources**
  Batch provisioning of multiple NECCI resources is possible by applying previously developed cloud template definitions as the inputs. The NECCI resources that can be input for provisioning include: the standard platform, the VMs, storage service, virtual LANs, firewalls, load balancers and monitoring settings of the high-availability platform. The cloud template is described in JSON, which is a text format featuring easy-to-understand syntaxes.
- **Multi-cloud**
  The NECCI provides two kinds of IaaS environment called the Standard Service (STD) and High Availability (HA) platforms. As the template-based provisioning enables the simultaneous building of a system across these IaaS environments, the efficiency of building a multi-cloud system that selects the IaaS environment to be used according to the server requirements can be improved.

3. Cloud-based SI Support Tool

Below, we describe the architecture and usage sequence of the cloud-based SI support tool that was prototyped based on the concept of the cloud SI in FY2014.

- **Architecture**
  Fig. 2 shows the architecture of the support tool, which is composed of a repository that saves the configuration information to be used as the model of the building target system at its center. Also stored are the CARDO and Alchemy data that function based on configuration information in the repository and of the template type provisioning function that manages the golden image generated by them in the template repository. The configuration information includes the AP load information required for sizing the target system and the configuration definitions specifying the product configuration and the settings of the system.
- **Usage sequence**

1) Model selection

This step selects the model of the building target system from the repository. The user selects the configuration

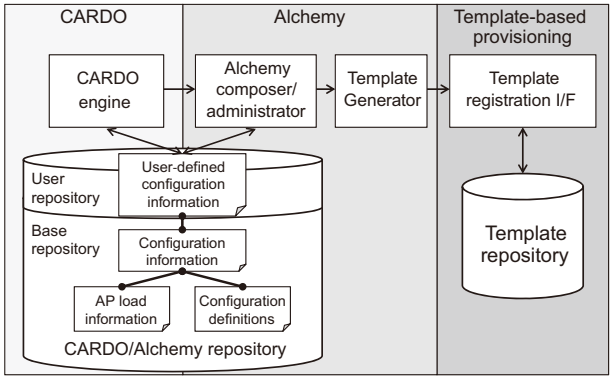

Fig. 2 Architecture of the cloud-based SI support tool.

information corresponding to the building target system from the repository and duplicates it for the project in question. The user can also designate a new piece of configuration information by combining more primitive components stored in the repository.

2) Estimation

This step estimates the specifications and number of servers of the building target system. The specifications and number of VMs to be allocated to the building target system are estimated automatically by CARDO influenced by the configuration information selected in step 1, the service level required by the customer and the availability requirements.

3) Customization

This step customizes the settings and configurations according to the requirements of various tests. Using the configuration definition in the configuration information duplicated in step 1) as the configuration definition for the actual system, modifications to the configuration definition for the production environment of each test scenario are defined. Customization for these tests is performed manually by the user and is dependent on each project.

4) Testing

This step builds the system execution environment for a specific test scenario and executes testing. The required number of VMs are activated for building the test environment according to the results of the estimations in step 2). The activated VMs are the ones for which only the intervention of the OS and Alchemy are installed. Following this, Alchemy is activated using the configuration definition customized for the test scenario in step 3) as the arguments and the system execution environment of the test target is deployed automatically in the activated VMs.

5) Release

This step outputs the cloud template of the tested system execution environment by means of Alchemy and registers the golden image of the building target system in the template-based provisioning.

It is expected that the sequence above will reduce the work period from the estimation to the template generation by 20% to 50%, though this figure will be dependent on the complexity of the applied system.

4. Conclusion

In this paper, we describe the concept of the cloud-based SIs for improving the efficiency and agility of the sizing, testing and the production environment building processes. The key technologies supporting cloud-based SIs including the CARDO model-based design support technology, Alchemy system configuration management technology and template-based provisioning are also discussed. In the future, we will pursue further R&D aimed at the rapid implementation of cloud-based SI technologies that can provide high-quality technologies to support our customers.

* Java is a trademark and a registered trademark of Oracle Corporation and/or its affiliates in the U.S. and other countries.

Reference

1) Chef:
https://www.chef.io/chef/
2) Puppet:
https://puppetlabs.com/
3) Service Component Architecture (SCA):
http://www.oasis-opencsa.org/sca

Authors' Profiles

SHIMAMURA Hisashi
Principal Researcher
Knowledge Discovery Research Laboratories

YANOO Kazuo
Principal Researcher
Knowledge Discovery Research Laboratories

KAJIKI Yoshihiro
Senior Expert
C&C Cloud Infrastructure Strategy Division

KURODA Takayuki
Assistant Manager
Knowledge Discovery Research Laboratories

NAKANOYA Manabu
Knowledge Discovery Research Laboratories

Information about the NEC Technical Journal

Thank you for reading the paper.
If you are interested in the NEC Technical Journal, you can also read other papers on our website.

Link to NEC Technical Journal website

Japanese

English

Vol.9 No.2 Special Issue on Future Cloud Platforms for ICT Systems

Vol.9 No.2
June, 2015

Special Issue TOP

NEC Information