FEATURE-MODEL-BASED COMMONALITY AND VARIABILITY ANALYSIS FOR VIRTUAL CLUSTER DISK PROVISIONING

Nayun Cho, Mino Ku, Rui Xuhua, and Dugki Min*

Department of Computer, Information & Communications Engineering,
Konkuk University, Seoul, Korea
{nycho,happykus,abealasd,dkmin}@konkuk.ac.kr

ABSTRACT

The rapid growth of networking and storage capacity allows collecting and analyzing massive amount of data by relying increasingly on scalable, flexible, and on-demand provisioned large-scale computing resources. Virtualization is one of the feasible solution to provide large amounts of computational power with dynamic provisioning of underlying computing resources. Typically, distributed scientific applications for analyzing data run on cluster nodes to perform the same task in parallel. However, on-demand virtual disk provisioning for a set of virtual machines, called virtual cluster, is not a trivial task. This paper presents a feature model-based commonality and variability analysis system for virtual cluster disk provisioning to categorize types of virtual disks that should be provisioned. Also, we present an applicable case study to analyze common and variant software features between two different subgroups of the big data processing virtual cluster. Consequently, by using the analysis system, it is possible to provide an ability to accelerate the virtual disk creation process by reducing duplicate software installation activities on a set of virtual disks that need to be provisioned in the same virtual cluster.

KEYWORDS

Virtual Cluster Disk Provisioning, Feature Model-based Virtual Cluster Commonality and Variability Analysis

1. INTRODUCTION

Virtualization is one of the promising solutions to overcome the limitation of computing power using a flexible resource scaling mechanism [1]. There are various researches in this direction to analyze big data with large-scale computational clusters using the virtualization technique [2,3]. Unlike physical clusters, a virtual cluster (VC) has a set of several virtual machines that needs to be provisioned before running on virtualized physical hosts. There are two steps of VC provisioning: VC placement and VC disk provisioning. The VC placement is a key factor to optimize the utilization of virtualized physical hosts using effective scheduling algorithms of underlying computing resources, such as VCPU, memory, network bandwidth, and so on [4,5]. On the other hand, the VC disk provisioning creates a set of virtual disks depending on the demand of the requested virtual cluster. Creating a set of virtual disks is time consuming tasks.

David C. Wyld et al. (Eds) : CCSEA, CLOUD, DKMP, SEA, SIPRO - 2016
pp. 45–52, 2016. © CS & IT-CSCP 2016 DOI : 10.5121/csit.2016.60205

Therefore, the way of VC disk provisioning directly affects the quality of service of cloud provider [6].

The provisioning process starts with the installing system or application software, such as operating systems or middleware, on an empty disk image. Among these software, some of system or application software, are repeatedly requested by users to install the software on virtual disk images. If there are pre-installed virtual disk images in shared storage (e.g., distributed file system), then the virtual disk image can be reused with a cloning mechanism. The cloning method for the virtual disk provisioning significantly reduces time to create a set of virtual disks of a virtual cluster. However, finding cloneable virtual disks that fully meet the demands of software on a virtual cluster is not a trivial task.

In order to find such reusable virtual disks for the virtual cluster disk provisioning, we apply Software product line (SPL) [7] methodology. SPL is a solution to create a collection of similar virtual disk images from existing shared assets (e.g., virtual disk images) by commonality and variability analysis of the product. To describe commonality and variability, this paper employ Feature model (FM) [8] as a metadata of a virtual disk image. FM is a hierarchical representation model that organizes commonality and variability of all the products of the SPL using features and their relationships. Applying FM to a virtual disk image enables disk provisioning system to determine which software features are commonly used in a virtual cluster. However, generating all the FMs related to a virtual cluster in a manual way is a tedious and error-prone effort. Consequently, a commonality-and-variability analyzer is necessary to generate the related FMs of the virtual cluster automatically.

This paper presents a methodology to provisioning a group of virtual disks for a virtual cluster in terms of software product line engineering. The virtual cluster disk provisioning process based on SPL, which includes (1) analyzing common and variant software features of a VC, (2) retrieving reusable virtual disk images, (3) generating virtual disk provisioning plan, and (4) creating virtual disk images. However, among these provisioning phases, this paper focus on the VC commonality and variability analysis with a case study. In order to analyze common and variant software features of a virtual cluster, several functions are needed. Firstly, the basic structure of feature model for a virtual disk image should be defined. Secondly, feature models of virtual disks for a VC should be generated according to the user's requirements. Thirdly, categorizing the type of virtual disk images should be performed automatically for correctness.

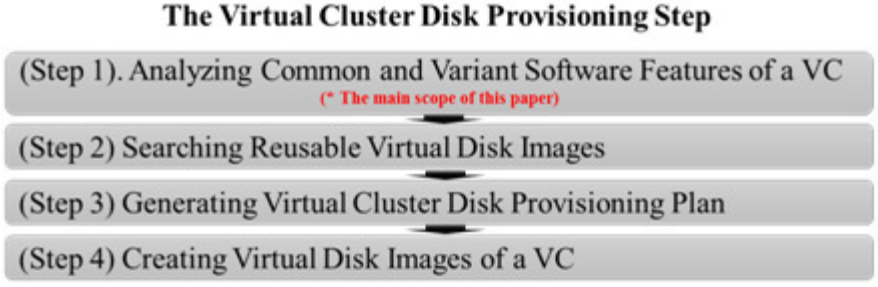

Figure 1. The Disk Provisioning Step of a Virtual Cluster

In this paper, we present the analysis system, named VC C&V analyzer, which archives the aforementioned functional requirements by a feature model reasoning mechanism. At first, VC C&V analyzer generates feature models a VC subgroup by using the VC provisioning specification extended from Open Virtualization Format (OVF). After that, VC C&V analyzer merges the generated feature models to extract the common and variant software features of a given virtual cluster. By using the common and variant software features, VC C&V analyzer generates the final VC commonality and variability feature models of a virtual cluster to classify the types of the disk images that need to be provisioned. Finally, the automated support of the VC C&V analyzer allows to reduce the effort needed to create a set of similar virtual disk images and their similarity investigation.

The remainder of this paper is organized as follows: Section 2 describes the architecture and processing flow of feature model-based VC commonality and variability analyzer with a feasible case study in Section 3. Section 4 discusses related researches and finally, Section 5 presents concluding remarks.

2. VC COMMONALITY AND VARIABILITY ANALYSIS METHOD BASED ON FEATURE MODEL

This section presents a commonality and variability analysis method based on the feature model for virtual cluster disk provisioning. Figure 2 shows a virtual cluster for our case study. Normally, a virtual cluster consists of a set of VC subgroups, such as Hadoop VC subgroup and HBase VC subgroup. Also, each VC subgroup is composed of virtual machines with the same system or application software. In this example, six virtual disk images should be provisioned for a big data processing VC.

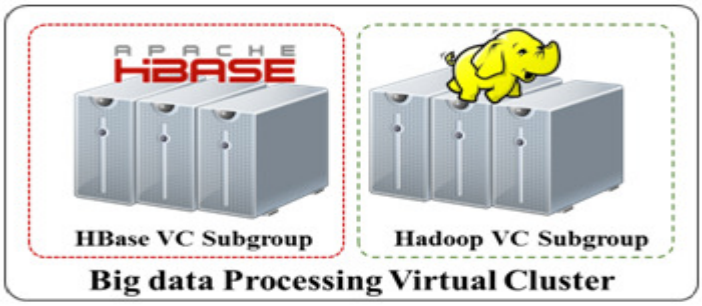

Figure 2. The Big Data Processing Virtual Cluster

In some cases, a set of VC subgroups of a virtual cluster may use similar software platform, such as operating system. For example, Hadoop VC subgroup and HBase VC subgroup may require same system software, named Debian Linux. Thus, commonality and variability analysis among the subgroups of a VC should be done to avoid duplicated installation tasks for the same software platform on a virtual disk. In order to analyze commonality and variability between VC subgroups, the basic structure of feature model for a virtual disk image is needed. Feature model allows the virtual disk provisioning system to categorize types of virtual disks which should be provisioned.

For provisioning the big data processing VC, the virtual disk provisioning system should classify types of virtual disks of Hadoop VC subgroup and HBaseVC subgroup that contain the same

software, such as Debian Linux, or different software, such as hadoop or hbase. Also, the dependencies between software and system architecture, such as AMD64 or i386, should be considered. To support these requirements, we define the basic structure of the feature model that includes architecture, system software, and application software. System software consists of various distributions and each distribution contains its own version. For example, there are several distributions of Linux operating system with version, such as Debian 8.0, Ubuntu 14.04, and CentOS 5.0. According to these types of the distribution, there is a set of variations that determines which packages to be installed in the system distribution, such as minbase, base, buildd, and so on. Similarly, application software consists of name, version, and variants

Since feature model presents commonality and variability of relevant products itself, the VC C&V analyzer generates feature models of VC subgroups using OVF-based virtual cluster specification which defined by the user. The VC C&V analyzer uses the specification as a requirement to meet the needs of a particular purpose of the virtual cluster. This specification involves a virtual hardware specification, such as the number of VCPUs, the size of memory and disks, the virtual network bandwidth required for each virtual machine, and name and version of software of each virtual cluster named *VirtualSystemCollection*. The VC C&V analyzer travels the *VirtualSystemCollections* to extract system and software information from *OperatingSystemSection* and *ProductSection* of VC subgroups. *OperatingSystemSection* involves architecture, distribution, variant, and version of the system software with attributes named id and version. *ProductSection* presents the name and version of application software which the provisioning engine needs to install in a set of virtual disks. Moreover, the end user can describe the relevant software configuration in this section, such as IP address, configurations regarding with a particular application software, and so on.

Using this information of VC subgroups, the VC Subgroup FM Generator (VC Subgroup FMGen) of the VC C&V analyzer maps an architecture variable into the Architecture feature, and name, variant, and version variables into the Distribution and Variant features respectively. Similarly, the VC Subgroup FMGen maps variables of the name, version, and variant of application software to Application feature.

After generating feature models of each VC subgroup, the VC Commonality and Variability(C&V) Feature Model Generator (VC C&V FMGen) merges the generated feature models to analyze the commonality and variability between VC subgroups. If there is only one VC subgroup in the provisioning specification, the VC C&V analyzer skips this step. In our research, FAMILIAR [9] is used for this merging step. In other words, the core features of the merged feature model can be interpreted as common features between VC subgroups, whereas different features can be defined as the various features of the VC subgroups. Using “merge” function of the FAMILIAR, VC C&V FMGen recursively combines the feature models of VC subgroups and generates a VC C&V feature model of the whole VC.

The generator performs “cores” and “mergeDiff” functions to divide the merged feature model into the VC commonality feature model and the VC variability feature model. Some of these feature models contain more than one product (e.g., virtual disks) and some of them involve an individual product of a virtual disk. Since we employ a feature model to describe the metadata of each virtual disk, the generator splits the model to produce a specific type of virtual disk in a case of the feature models that include more than one product. Consequently, VC C&V analyzer

generates final results including a list of locations of the generated feature models and the number of related VC subgroups of each feature model.

3. CASE STUDY

This section presents a case study of applying the proposed VC C&V analysis method to a big data processing VC. Basically, Hadoop and HBase requires master and slave nodes to handle big data in a distributed way. To avoid the performance degradation under hot spotted case [10], we separately design VC subgroups along with the big data processing middleware (e.g., Hadoop and HBase). Figure 3 shows the detailed description of a virtual cluster including Hadoop and HBase subgroups.From now, this section describes four processing steps to analyze commonality and variability of a big data processing VC by using the VC C&V analyzer.

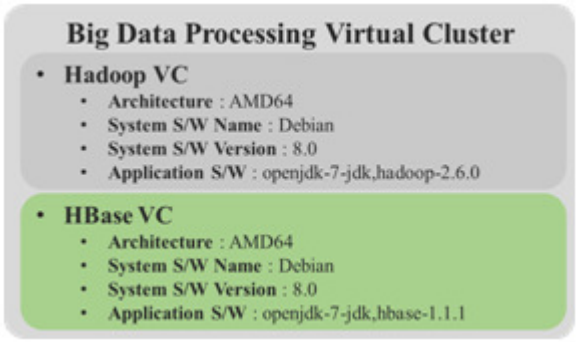

Figure 3. A Case Study of the Big Data Processing Virtual Cluster

Step 1 – Extracting architecture, name, variant and version of the system and application software: Firstly, the VC C&V Analyzer extracts the architecture, name, version, and variant of system and application software of the big data processing virtual cluster as shown in Table 1. The result can be categorized by the name of each VC subgroup. Consequently, the extracted information of system and application software is imported into the VC subgroup Feature Model Generator (VC Subgroup FMGen).

Table 1. The extracted information of system and application software of each VC subgroup.

VC Subgroup Name	Software Type	Software Name	Software Version	Software Variant	Architecture
Hadoop VC	System Software	Debian	8.0	base	amd64
	Application Software	Openjdk	7.0	-	amd64
	Application Software	Hadoop	2.6.0	-	amd64
HBase VC	System Software	Debian	8.0	base	amd64
	Application Software	Openjdk	7.0	-	amd64
	Application Software	HBase	1.1.1	-	amd64

Step 2 – Generating VC subgroup feature models: The VC subgroup Feature Model Generator in the VC C&V analyzer maps the resulting information of system and application software to the basic structure of feature model. Figure 4 shows a snippet of the generated Hadoop and HBase VC subgroup feature models. By using these feature models, VC C&V Analyzer performs the

feature model comparison to determine which software features are commonly used among the VC subgroups.

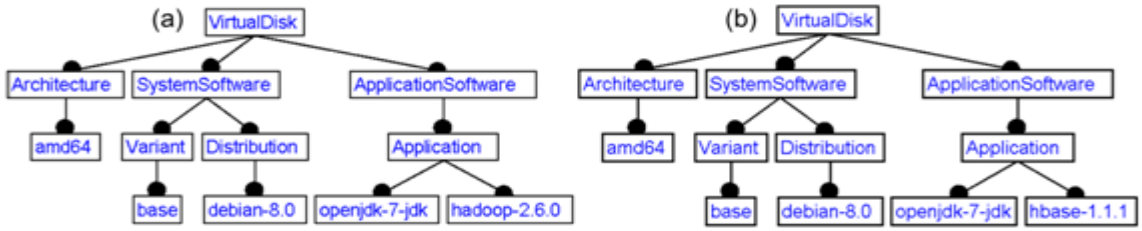

Figure 4. The generated feature models of Hadoop (a) and HBase(b) VC subgroups

Step 3 – Comparing feature models of each VC Subgroup: It is easy to categorize commonality and variability of the resulting feature models in manual. However, to automate such analysis activities, we employ a method of feature model reasoning using FAMILIAR framework. Figure 5 shows the consequent results of step 3.

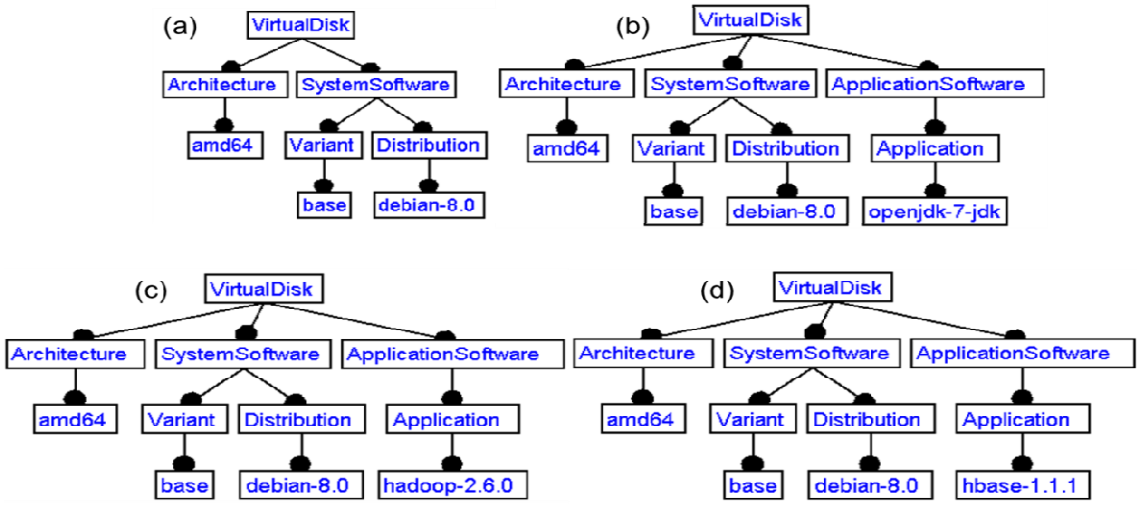

Figure 5. The comparison results between Hadoop and HBase VC subgroups.
5(a) and 5(b) presents common software features, and 5(c) and 5(d) indicates different software features among the VC subgroups.

As shown in Figure 5, VC C&V Analyzer generates feature models of commonality and variability based on Hadoop and Hbase VC subgroups. Also, each feature model indicates an individual type of virtual disk, such as amd64 architecture based debian-8.0, openjdk-7-jdk, hadoop-2.6.0, and hbase-1.1.1. These results will be used on the next virtual disk provisioning step to investigate a reusable virtual disk from the reusable asset repository.

Step 4 – Generating final results of the VC C&V Analysis: As a final result of the VC C&V analyzer, we employ a JSON-based result model. The final result is generated by two steps. Firstly, it categorizes a set of groups which use shared virtual disks among the virtual machines. In this case study, we design none of VC subgroups share virtual disks with other virtual machines. Secondly, it describes required quantity and location of the commonality and variability feature models. From the final result, it is easy to determine which reusable virtual disk meets the analyzed types of virtual disks or not.

4. RELATED WORK

This section presents some efforts in the area of virtual disk image provisioning in context of SPL [7, 11, 12, 13]. Among these researches, Wittern, Erik, et al. [11] present an Infrastructure-as-a-Service (IaaS) deploy model to describe IaaS consumer requirements, including VMs, virtual disk images, and software installed on the images using feature model. Once IaaS consumer selects the cloud provider, VM type, and virtual disk image for the VM, deployment engine invokes a web service call to instantiate VM described in the selected IaaS deploy model. After instantiate VM, the software installation tasks are executed via SSH using a configuration management tool, such as Chef. Also, Dougherty, et al. [12] shows an approach to optimizing configuration and cost of auto-scaling cloud infrastructure. They provide a feature model of virtual machine configuration that captures software platform, including operating system and applications.

Similar to the aforementioned research, the configuration of cloud infrastructure is generated by a selection of features from the feature model in a manual way. Using the configuration, they aim to find a matching virtual machine that already pre-booted in the auto-scaling queue. Krsul, Ivan, et al. [13] provides a direct acyclic graph-based model for configuration activities of a VM. If there a partial graph matching with a set of graphs stored in the repository, named VM shop, the system configures the partial matches of cache VM images as follow as the production line which controls procedures for cloning and configuring a VM. There are several works to employ SPL to create images for a virtual machine, however, none of the works has been addressed how effectively SPL can be used for provisioning virtual disk images of a virtual cluster.

5. CONCLUSION

This paper described a way to provisioning virtual disk images of a virtual cluster via feature model-based VC description and their commonality analysis. We presented our methodology in the context of a feature model-based commonality and variability analysis of a VC that provides an ability to accelerate the provisioning process by reducing duplicate type of virtual disks in the same virtual cluster.

We presented detailed processing flow of the VC C&V Analyzer to determine which types of virtual disks should be provisioned together in a given virtual cluster. We have applied VC C&V Analyzer to investigate common and variant types of virtual disk images among VC subgroups of big data processing. Moreover, our experience in using the VC C&V Analyzer to generate a feature model of each VC subgroup and compare these resulting feature models to determine software which need to be provisioned commonly in this case study. There are still remaining important issues concerning VC disk creation by using the result of VC C&V analysis. We are addressing these remaining challenges as part of our future work.

ACKNOWLEDGEMENTS

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2015-H8501-15-1011) supervised by the IITP(Institute for Information & communications Technology Promotion).

REFERENCES

[1] Foster, I., Zhao, Y., Raicu, I., & Lu, S., (2008) "Cloud computing and grid computing 360-degree compared. In Grid Computing Environments", Workshop, GCE'08, pp. 1-10. IEEE.

[2] Foster, I., Freeman, T., Keahy, K., Scheftner, D., Sotomayer, B., & Zhang, X., (2006) "Virtual clusters for grid communities. In Cluster Computing and the Grid", CCGRID 06. Sixth IEEE International Symposium on Vol. 1, pp. 513-520. IEEE.

[3] Hoffa, C., Mehta, G., Freeman, T., Deelman, E., Keahey, K., Berriman, B., & Good, J., . (2008) "On the use of cloud computing for scientific workflows" In: eScience, eScience'08. IEEE Fourth International Conference on, pp. 640-645. IEEE.

[4] Sotomayor, B., Montero, R. S., Llorente, I. M., & Foster, I, (2009) "Virtual infrastructure management in private and hybrid clouds. Internet computing", Vol. 13, No. 5, pp. 14-22. IEEE.

[5] Mino Ku, (2015) "Flexible and Extensible Framework for Virtual Cluster Scheduling", Ph.D. Thesis, Konkuk University.

[6] Juve, G., & Deelman, E.: Wrangler, (2011) "Virtual cluster provisioning for the cloud", In Proceedings of the 20th international symposium on High performance distributed computing, pp. 277-278. ACM.

[7] Pohl, K., Böckle, G., & van der Linden, F. J, (2005) "Software product line engineering: foundations, principles and techniques", Springer Science & Business Media.

[8] Kang, K. C., Lee, J., & Donohoe, P, (2002) "Feature-oriented product line engineering", IEEE software, Vol. 4, pp. 58-65, IEEE.

[9] Acher, M., Collet, P., Lahire, P., & France, R. B, (2013) "Familiar: A domain-specific language for large scale management of feature models", Science of Computer Programming, Vol. 78, No. 6, pp. 657-681.

[10] Wu, Y., & Gong, G, (20 A Fully Distributed Collection Technology for Mass Simulation Data. In Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on, pp. 1679-1683. IEEE. (2013)

[11] Wittern, E., Lenk, A., Bartenbach, S., & Braeuer, T, (2014) "Feature-based Configuration of Vendor-independent Deployments on IaaS", In Enterprise Distributed Object Computing Conference (EDOC), IEEE 18th International, pp. 128-135. IEEE.

[12] Dougherty, B., White, J., & Schmidt, D. C, (2012) "Model-driven auto-scaling of green cloud computing infrastructure", Future Generation Computer Systems, Vol. 28, No. 2, pp. 371-378.

[13] Krsul, I., Ganguly, A., Zhang, J., Fortes, J. A., & Figueiredo, R. J, (2004) "Vmplants: Providing and managing virtual machine execution environments for grid computing", In Supercomputing, 2004. Proceedings of the ACM/IEEE SC2004 Conference, pp. 7-7. IEEE.