An approach of VM image customized through Linux from scratch on cloud platform

Gaochao Xu[1, 2], Yushuang Dong[1*], Bingyi Sun[1], Xiaodong Fu[1], Jia Zhao[1]

[1] *Department of Computer Science and Technology, JiLin University, Changchun 130012, China*

[2] *Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China*

Received 1 March 2014, www.tsi.lv

Abstract

The cloud platform provides abundant resources and services for users. More and more mobile users began to use the cloud services. They have higher real-time demands on service. The size of traditional virtual machine (VM) operating system is basically large. It will take many resources in deployment and communication process, and always affect the real-time performance of system. To reduce communication overhead and improve deployment speed of VMs, this paper proposes an approach of customized VM image with LFS. LFS can reduce the size of VM image efficiently and enable flexible customization of the VM image by incremental customization. The experimental results show us that the size of VM image generated by the proposed method is smaller than the one generated by kernel tailoring technology in system overhead. Meanwhile it is also faster in running speed.

Keywords: Cloud computing, Linux from scratch, Customized virtual machine

1 Introduction

Cloud computing is at the forefront of the information technology. With the development of mobile devices and the increasing requirement of mobile user, cloud computing begin to be applied in providing service for mobile users. The size of traditional VM image is so large that it has a profound impact on real-time. If the size of VM image could be decreased, we can realistically reduce system overhead and communication overhead for cloud platform. With cloud computing application development on mobile platforms and other devices, the micro-kernel technology will be more demanded under cloud computing environment. This paper will analyze how to get customized VM image through LFS. The full name of LFS is "Linux from Scratch" [1]. LFS refers to building a Linux system manual or an idea rather than a release version of Linux. Unlike ordinary Linux installation, LFS guides to compile the open-source software packages into a needed smallest, fastest Linux system through the host system. Users can control all features of the new system during the build process such as Installation Directory and File organization form, parameters and permissions settings, etc. We can improve the real-time performance effectively by using the customized system.

In section 2 of this paper, we will give a brief introduction of related work about current VM image customized methods. In section 3, the design and implementation of customize VM image through LFS will be presented in detail. Than in section 4, experiments undertaken and results obtained will be shown which demonstrate that the new method provides an effective solution. Finally, we will conclude the paper in section 5.

2 Relevant work

Because of the large size of the traditional VM image, the real-time performance of system is poor and too many system resources are employed. It becomes necessary to reduce the system size. Although many Linux kernel-cutting methods can efficiently decrease the size of Linux kernel, these methods are usually used to customize the Linux kernel for embedded system [2, 3] and many other specific fields [4]. These methods are not appropriate for cloud computing and micro-kernel technology is not yet universally applied in the cloud computing.

Application of virtualization technology proposed software as a service model. As App-V [5] of Microsoft, ThinApp [6] of VMware and Citrix XenApp [7], individual does not need to consider the process of installation, maintenance and upgrade. These operations can be completed by software service providers. Users obtain the right to use the software via user identity authentication mechanism. They distinguish the application and the OS, the centralized management, the maintenance and upgrading of the software. Software provider on-demand provides the software for user; users can use them without installation. However, most of the proposed plans are business plans for windows, and their application needs to run with network support. Therefore, OpenAppV [8] proposed on-demand Customized Virtual Machine Instance System. These methods realized software customization, but did not reduce the system

* ***Corresponding author*** e-mail: yushuangdong@gmail.com

size. In general, VM developed by the traditional methods is often difficult to be understood and modified. It is not satisfied enough in extensibility and reusability. It is very difficult to dynamically change and extend the VM. It needs frequently to full-manually override the existing VM [9, 10]. Reference [11] proposed An Implementation Approach to Custom-Built Virtual Machines. However, this method is relatively slow and not suitable for high real-time systems. To reduce the system size and to improve the real-time, we will analyze getting the customized VM image through LFS in this paper.

3 VM image customized with LFS

At present, jhalfs can realize the automatic installation of LFS by extracting the command from the XML of lfsbook. jhalfs can choose flexibly whether LFS process tests the installation and optimizes the system etc. by setting common/config file under the jhalfs. In order to meet users' requirements, we need generate customized VM image with the minimum matching image incremental installation, for which, we need to get the configuration information of users' requirements through the management process of the cloud platform. Therefore, this paper analyzes how to realize automatic installation of LFS, and the application software and services based on our own shell scripts.

GCC is a GNU [12] compiler containing C / C + + compiler, which can compile the source code program to generate the target file in combination with other basic tools. Binutils [14] is a set of binary tools including connectors, assembler and other tools used in the target file and it can transform the target file into an executable file. Make is a program similar to the shell script, which can control the entire compile-link process through reading a makefile that contains the source code and document library dependencies and rules. Glibc, the crucial link in the production process of LFS, is also very important for Linux. All dynamic linker must use it. In the process, aforementioned four tools are dependent on the C Runtime library glibc.

The toolchain is a temporary build environment. It is used to generate the target system that includes all necessary tools such as GCC and binutils. Their versions have exactly the same as the tools of the target system.

As the Linux kernel source code we want to compile requires a specific toolchain and glibc version and an environment that are inconsistent with the host system. Therefore, we have to first build a specific toolchain and glibc to get the target system. For this purpose, all the tools on the toolchain are compiled by their respective source codes. GCC, binutils, make, common tools in the compilation process are dependent on glibc. Similarly, the glibc needed is a particular one rather than the one in the host system. This particular glibc is compiled by another tool chain, which we call a temporary toolchain. While this temporary toolchain is compiled by host systems obtained through a set of own compilation tools, glibc, other C libraries and Linux kernel, etc.

The implementation process of LFS shown in Figure 1:

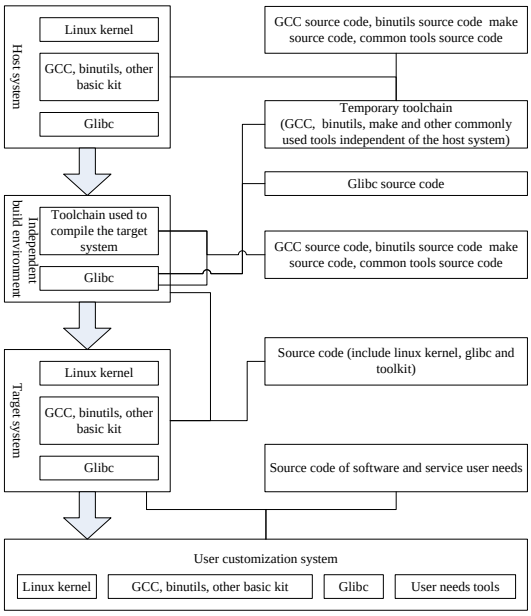

FIGURE 1 LFS implementation process

1) Temporary tool chain compiled by source code of the tool set on the host system;

2) Get an independent glibc library by compile the glibc source code using the temporary tool chain?

3) Use the independent glibc library in the host system environment to build independent toolchain, independent build environment completed;

4) Compile the Linux source code package in the independent build environment and build the Linux kernel for the operating system.

5) Get the user request information and compiled required software and services in the target system according to user requests. Then generate customized system image.

6) Finally, do some final changes work as removing the source package, temporary toolchain, and independent build environment, and restore the original system parameters and configuration files.

The process of customizing VM image as showed in Fig. 2:

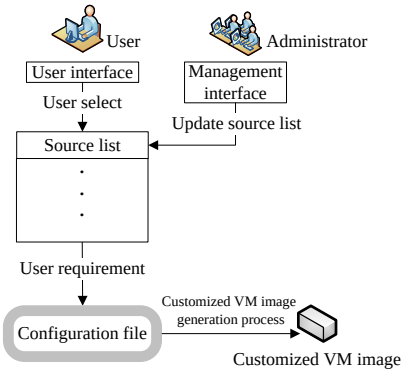

FIGURE 2 LFS implementation process

3.1 USER INTERFACE

It is easy to access and operate by using the interactive interface of traditional web, and gives a list of application software and services in a platform. User can log in the platform for selecting their needs through a user interactive interface, as showed in Fig. 3. Sent through the network management process to the cloud platform, and user request information is stored in the management side.

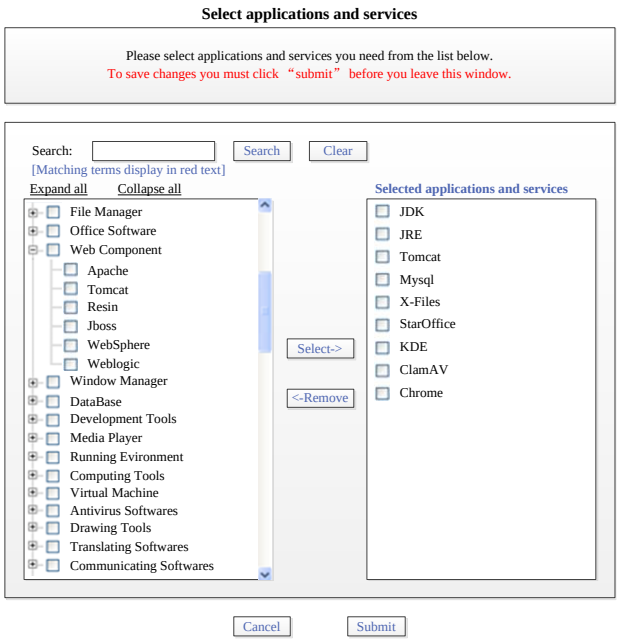

FIGURE 3 User interface

Shown in Fig. 4, the applications and components provided by the management side of the cloud platform also can be dynamically updated when new applications or components need to be updated to the cloud platform. Administrators use an interface to upload applications or components. When the list of components is missing from the system components that applications relied on, it needs administrators upload the missing system components and the management process generated script code used to complete the installation of the customization process. Administrators of cloud platform can dynamically update or delete the existing applications and components. Management process records the information when administrators upload applications and components. The information mainly includes five parts: (a) id of application or component, (b) name of application or component, (c) size of application or component, (d) components information that application relied on, (e) installation script information.

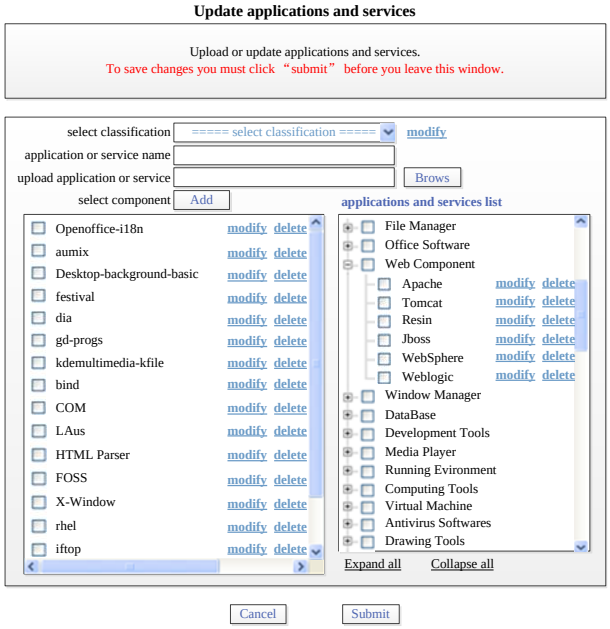

FIGURE 4 Applications and services management interface

3.2 CONFIGURATION FILE

Users select their needs through user interface and send the request information to the management process. The management process receives the request information and extracts it to form a customized XML configuration file. The information of the configuration file mainly includes three parts: (a) hardware requirements include CPU, hard disk, memory, and number of computing nodes, (b) applications requirements, (c) components requirements. We use id to record applications and components that user requested in configuration file. Configuration file fragment is shown as follows:

```
1: <user-request>
2:   <hardware name="userid">testuser</hardware>
3:   <hardware name="vmnum">20</hardware>
4:   <hardware name="cpu">1</hardware>
5:   <hardware name="ram">1024</hardware>
6:   <hardware name="disk">80</hardware>
7:   <application name="id">2, 6, 8, 9, 11</application>
8:   <component name="id">4, 7, 14</component>
9: </user-request>
```

Lines 3-6 record hardware requirements include the size of virtual cluster, frequency of VM CPU, memory size of VM and disk size of VM. Lines 7-8 record the id of applications and components that satisfy the user requirements.

3.3 CUSTOMIZED IMAGE

The management process extracts the information I_{conf} from the configuration file. The information of I_{conf} includes the user request. According to I_{conf}, user needs K

VM nodes. There are N exist VM image copies stored on cloud platform. The VM image copy is used to generate VM image that users customized previously. The initial minimized image is P_0. The match degree is calculated by matching configuration file information of VM image copy with I_{conf}. The process traverses the copy and matches the copy of node c ($0 =< c < N$) with I_{conf}. If the copy node c exists other applications or services beyond I_{conf}, the node cannot be a matching node. If the copy of node c does not exist other applications or services and the match degree is greater than the former one, then $P_{target} = P_c$. If the copy node c matches exactly with I_{conf}, then $P_{target} = P_c$, now finish the traversal process to get customized VM image P_{target}. If there is not an exactly matching node, we need to completely traverse the copy in order to get P_c with the largest match degree to obtain customized VM image P_{target} and to install the application software and services in I_{conf} that P_{target} does not have to get customized VM image P_{target}. Then we store it in a virtual node and update the copy of the image to store this node information in that copy. The customized VM image generation process is as follows:

1: $P_0 \leftarrow$ initial VM image copy
2: $S_0 \leftarrow$ size of P_0
3: $I_0 \leftarrow$ configuration file information of P_0
4: $M_0 \leftarrow$ match degree of P_0
5: $P_{target} \leftarrow$ target VM image copy
6: $S_{target} \leftarrow$ size of P_{target}
7: $M_{target} \leftarrow$ match degree of P_{target}
8: $P_{target} = P_0$, $S_{target} = S_0$, $M_{target} = M_0$
9: $c = 0$
10: While $c < N$ do
11: If I_c exactly matches with I_{conf}
12: $P_{target} = P_c$
13: Break
14: Else
15: If I_c exist other applications or components information beyond I_{conf}
16: $S_c = 0$, $M_c = -1$
17: Else
18: If $M_c > M_{target}$
19: $P_{target} = P_c$, $S_{target} = S_c$, $M_{target} = M_c$.
20: End if
21: If $M_c = M_{target}$ and $S_{target} < S_c$
22: $P_{target} = P_c$, $S_{target} = S_c$, $M_{target} = M_c$
23: End if
24: End if
25: End if
26: $c = c + 1$
27: End while
28: If $c == N$
29: $P_{temp} \leftarrow$ According to I_{conf}, complete the P_{target} installing
30: $P_{target} = P_{temp}$
31: End if
32: store the P_{target} and I_{target} on cloud platform, set the host as target host
33: generate customize VM image through P_{target}, create K VM nodes

Lines 1-4 give VM image copy P_0 as input that has no extra applications or components. P_0 is generated by LFS process. Lines 5-8 initialize the target VM image copy P_{target} as output. Lines 10-27 find the target VM image copy that mostly satisfies the user's requirements. If the VM image copy exists other applications or components beyond the user's requirements, it is not the VM image we need. If there are two or more VM image copies mostly satisfy the user's requirements, we should compare the size of these VM image copies and choose the biggest one as the target VM image copy. Lines 28-30 show that if there is no VM image copy which exactly satisfies the user's requirements, we can get a VM image copy that satisfies most the user's requirements, then according to user's requirements, complete the target VM image copy installation and get a new target VM image copy. Lines 32-33 store the target VM image copy on cloud platform and set the host that stored the target VM image copy as target host. Finally we generate customize VM image through the target VM image copy and create K VM nodes that satisfy the user hardware requirements.

4 Experiment analysis

To establish a minimum system requires about 1.3GB of the partition so as to have enough space to store and compile all the source packages. A larger space (2~3GB) is needed to install the application software and services that the user needs. The LFS system itself does not occupy so much space and most of the space required is used to provide adequate temporary space for the software compiler. It takes many temporary spaces to compile the package. However, these temporary spaces can be recycled after software installation done. We would better to use a small hard disk partition as swap space because memory (RAM) is not always enough during the compilation process. The kernel uses swap space to store the data in order to free up memory space for running processes. The swap partition that LFS system uses can be the same with the one that the host system uses. Therefore, we do not have to create a new one for the LFS system when the host system already has a swap partition. In this paper, we suggest getting customized VM image through LFS in order to achieve the goal of reducing system and communication consumption.

Experiment Environment: In our experimental configuration, hosts with the same type are selected. We use HP proLiant ML350 G6 (AU662A) as hosts in cloud platform. These hosts are configured with Xeon E5506 2.13GHz four core processor, 8GB DDRIII RAM, 4TB 7.2K 6Gbps hard disk and NC326i PCI Express 1000Mbit/s NIC. In order to simplify the process of customized VM image generation, we install LFS Live CD 6.2-3 with kernel 2.6.16.26 on all hosts as host system. Virtual Tool is Xen 4.1.1. We configure all VMs with single core, 40GB VM hard disk. To ensure parent-

VM boot successfully, we configure VM memory size with 512M.

Experiment: To reduce system resource occupation and to fundamentally solve the problem of too long virtual machine downtime in the deployment process, the VM image size should be as small as possible. We compare the customized VM image generated by LFS with current lite release version of Linux.

Experimental comparison results are showed in Figure 5, 6. We customize VM image as web servers. We can limit the size of customized VM image generated by LFS at 38.63M. Comparing with other lite release version of Linux, LFS visibly reduced the system consumption, simplified customize process, and it is much easier in the customize process. It also has advantage in booting speed and system consumption. We configure VM memory size with 512M. LFS takes only 6.76s to complete the booting process. The boot speed of VM that load customized VM image generated by LFS is faster than others. By reducing the VM image size, we effectively reduce the VMs communication consumption.

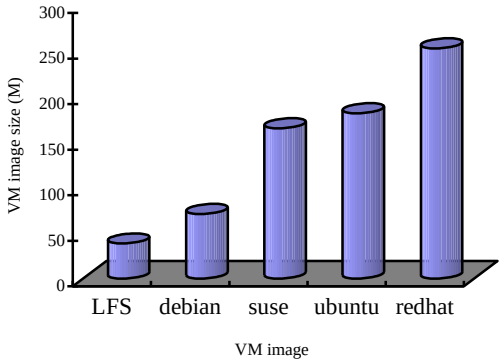

FIGURE 5 System space occupation comparisons

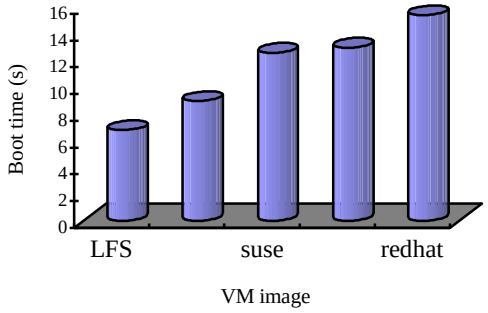

FIGURE 6 System booting time comparisons

5 Conclusion

Cloud computing has a promising development prospects and the related key technologies are growing rapidly. In this paper, we give a summary of the existing VM image customized technology and present the design, implementation and evaluation of customized VM image through LFS. We customize the VM image with LFS in order to reduce the VM image size, thus decrease the system overhead and communication overhead. The customized VM satisfies the user's request and consume less space.

To further improve the performance of VM image customized, there are still many problems that need to be solved in the future. In customized VM image generation process, it traverses the copy to find the target host. It is not suitable for high real-time demanded cases. We plan to design a faster matching method to find the target host. We find that the target host takes only the match degree in consideration. In a cloud platform with workload or in a heterogeneous cloud platform, it is not enough and the target host we find may be not the best one. We plan to add the performance parameter for finding the target host.

References

[1] Beekmans G 2007 *Linux From Scratch* http://www.linuxfromscratch.org/lfs/view/stable/
[2] Fröhlich A A, Schröder-Preikschat W 1999 Tailor-made operating systems for embedded parallel applications *Lecture Notes in Computer Science* 1361-73
[3] Hasan M Z, Sotirios S G 2008 Customized kernel execution on reconfigurable hardware for embedded applications *Microprocessors and Microsystems* 211-20
[4] Montgomery J, Brewster G B, Yee W G 2010 A customized Linux Kernel for Providing Notification of Pending Financial Transaction Information *7th IEEE Consumer Communications and Networking Conference* 1021-2
[5] APP-V [EB/OL]. http://www.microsoft.com/app-v
[6] ThinApp [EB/OL]. http://www.vmware.com/products/thinapp/
[7] XenApp [EB/OL]. http://www.citrix.com/xenapp
[8] Zhang Hanying, Wu Qingbo, Tan Yusong 2013 On demand Customized Virtual Machine Instance System *Computer Technology and Development* **23**(4) 1-10
[9] Ert l M A, Gregg D, Krall A, et al. 2002 Vmgen: A Generat or of Efficient Virtual Machine Interpreters *Software-Practice & Experience* **32**(3) 265-94
[10] Ert l M A, Gregg D 2004 The Structure and Performance of Efficient Interpreters *Proc of the 2004 Workshop on Interpreters, Virtual Machines and Emulators*
[11] Ouyang Xingming, Zhu Jinyin 2008 An Implementation Approach to Custom-Built Virtual Machines and Their Dynamic Optimization *Computer Engineering & Science* **30**(1) 129-41
[12] GNU Binutils http://en.wikipedia.org/wiki/GNU_Binutils

Authors

Gaochao Xu, born in 1966, Xiaogan City, Hubei Province, China

Current position, grades: Professor, Doctor
University studies: Computer Science and Technology
Scientific interest: Distributed System, Grid Computing, Cloud Computing, Internet Things, etc.
Publications: 55
Experience: Professor and PhD supervisor of College of Computer Science and Technology, Jilin University, China.

Yushuang Dong, born in 1983, Jixi City, Heilongjiang Province, China

Current position, grades: PhD
University studies: Computer Science and Technology
Scientific interest: Distributed System, Cloud Computing.
Publications: 8
Experience: PhD of College of Computer Science and Technology, Jilin University, China.

Bingyi Sun, born in 1991, Changchun City, Jilin Province, China

Current position, grades: Master Degree Candidate
University studies: Computer Science and Technology
Scientific interest: Distributed System, Cloud Computing.
Experience: Master Degree Candidate of College of Computer Science and Technology, Jilin University, China.

Xiaodong Fu, born in 1966, Changchun City, Jilin Province, China

Current position, grades: senior engineer
University studies: Computer Science and Technology
Scientific interest: Distributed System, Cloud Computing.
Publications: 14
Experience: senior engineer of College of Computer Science and Technology, Jilin University, China.

Jia Zhao, born in 1982, Changchun City, Jilin Province, China

Current position, grades: Doctor
University studies: Computer Science and Technology
Scientific interest: Distributed System, Cloud Computing.
Publications: 11
Experience: Doctor of College of Computer Science and Technology, Jilin University, China.