RC25196 (W1108-006) August 1, 2011
Computer Science

IBM Research Report

Virtual Machines with Sharable Operating System

Trieu Chieu, Hoi Chan
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Virtual Machines with Sharable Operating System

Trieu Chieu, Hoi Chan
IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY, USA

e-mail: {hchan,tchieu}@us.ibm.com

Abstract— *Virtualization technologies commonly known as Cloud model enable the execution of multiple virtual machine instances (VMs) with different operating systems (OSs) on the same physical host. Each VM instance functions independently as an isolated system with its own physical resources, OS copy and applications. There is only a limited number of currently available and widely used OSs used by most of the running VM instances; it is wasteful to store all the VM images with virtually the same common OS code. It is also inefficient in terms of performance and system resources utilization to virtually clone the entire image each time a new VM instance is provisioned. In addition, performing OS updates and patches are complicated, tedious and error prone since not only the stored images need to be updated, all the running VM instances must be properly refreshed. More importantly, faster provisioning of VM instances in respond to workload changes is critical to the successful operation of Cloud service providers. In this paper, we show our exploration work to address these performance issues by using a common, sharable operating system approach which provides run-time on-demand operating system components to individual VM instances in Cloud environment. This new approach allows optimized VM image storage, faster VM provisioning and efficient OS updates with minimum interruption.*

Keywords-Cloud VM, Cloud, patching, virtualization, OS

I. INTRODUCTION

With the widespread adaptation of virtualization technologies and Cloud Computing [1,2,3] services and as Cloud service providers [4] expand in size, the number of VM images, VM instances and physical hosts increases substantially, with resources scattering across multiple locations and even continents [5]. Fast and responsive VM instance provisioning [6], optimal image storage [7] and efficient software updates and patches [8, 9] are the major game-changers in terms of cost, performance and customer satisfaction. VM image storage [10] has been studied extensively, and many researches focus on efficient mechanisms such as removing I/O bottlenecks, fast image retrieval structure and efficient and optimized namespaces, etc. There is little emphasis on actual image size reduction from the point of view of the efficient use of the operating system (OS). As there are only a limited number of currently available and widely used OSs, and with the well-developed virtualization technologies in image management and VM execution, sharing the OS code from a common file system by other VM instances is a real possibility. The traditional way of provisioning a VM instance requires a copy of the entire VM image which includes the OS, the required applications and configuration files. Faster VM instances provisioning can be achieved by directly reducing the size of the VM image by extracting and sharing common components in the OS. In addition, performing expected and emergency OS updates and patches are complicated, tedious and error prone since not only the stored images need to be updated, all the running VM instances must be properly refreshed. Image storage, provisioning latency and software updates down time and reliability are the major factors which affect user experiences and Cloud service provider profitability. A question that almost every CIO and software architect would ask: How can innovative software techniques be used to enable better user experiences and increased provider profitability in the Cloud setting? To help achieve these objectives, we explore a new approach on the OS level by maintaining a common, sharable central OS image which provides on-demand OS functionalities (components) to individual VM instances dynamically in a Cloud environment. The production VM images will only keep proxies to some common, non-application and non-user specific OS components, OS code will be loaded from the central OS system (disk or cache) to the actual running VM instances on demand, analogous to the dynamic class loading mechanism of the Java Virtual Machine [11]. VM image size will be reduced since only OS component proxies are maintained in the VM images, with user specific configuration and applications, in contrast to the entire OS system. For centralized managed VM instances such as those in a large data center and with the appropriate VM instance management tools, certain OS patches and updates can be performed on the central OS system (and its original VM image) and subsequently, all changes will propagate to all of its associated VM instances after refresh. The VM instances management tools for OS patching and updates on a common shared OS is beyond the scope of this paper and will be addressed in subsequent papers.

The rest of this paper is organized as follows: Section 2 gives an overview of the architecture of a shared OS approach in Cloud setting and describes the OS component sharing mechanism. Section 3 shows an analogous system and possible design and implementation approach using the common Linux file sharing system. Section 4 uses the VMWare ESX server, its snapshot management mechanism and APIs as the basis to prove the feasibility and show the increased performance of the proposed shared OS approach in Cloud setting. Section 5 discusses issues related to the shared OS approach and possible future works and Section 5 concludes.

II. OVERVIEW OF ON-DEMAND OS COMPONENTS

As we have observed in a typical desktop or laptop computer, its limited main memory does not contain the entire copy of the OS but only the components for the functionalities that are currently or recently used when it runs [12]. OS components for the required functionalities are swapped in as needed and out when done. We apply basically the same principal to the running VM instances in the Cloud setting – common components of the OS are factored out and persistently stored and centrally maintained, these components are swapped in or out by the running VM instances dynamically on an on-demand basis. Since the number of types of common operating systems (e.g. Windows, Linux, UNIX, AIX) used in most of the VM instances is small, and for each of the OSs with the same version and up-to-date service patches, the components are virtually identical. Maintaining a server or file system for the common OS components for each of the common operating systems is relatively manageable with reasonable cost.

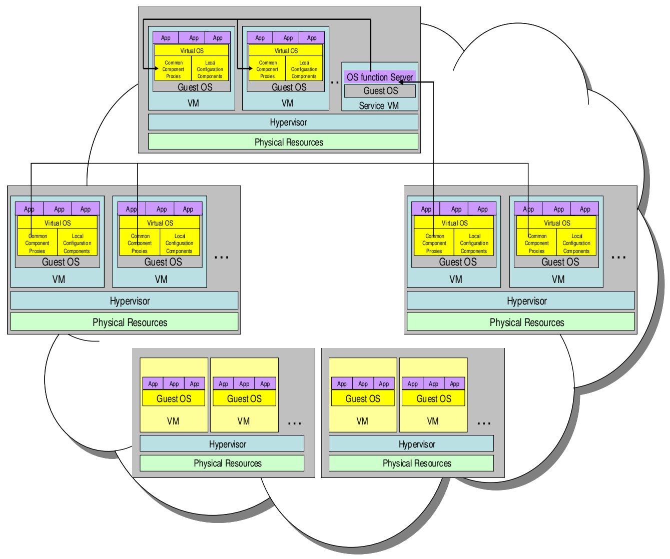

Figure 1. Cloud with VM OS component sharing

Figure 1 shows an overview of a typical Cloud environment with on-demand OS components sharing. In contrast to a classical Cloud setting, it maintains one or more dedicated VMs with an installed OS component server (Figure 2). Or alternatively, the OS components can be stored in a in a common sharable file system and accessed locally (Figure 3).

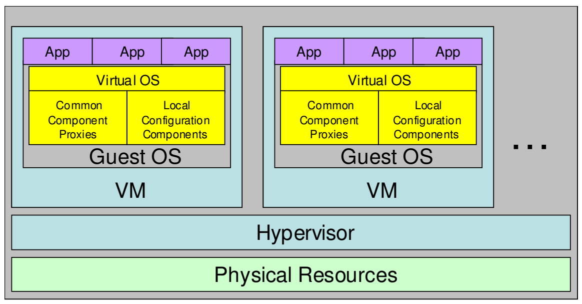

Figure 2. VM with thin virtual OS layer

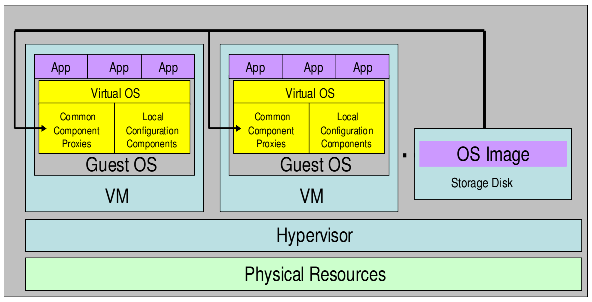

Figure 3. OS components from image storage

For simplicity, we will use the common OS server approach in subsequent discussions. The server hosts a collection of common, non-user and non-application specific OS components and will be delivered to any of its clients dynamically upon request (loaded into the client's main memory). For those VM instances that participate in the OS component sharing scheme, a thin version of OS of the required type (Figure 4) and a pointer to the OS server replaces the regular full OS copy. For those VM instances that do not participate in this scheme, it runs normally with the entire regular OS image stored locally.

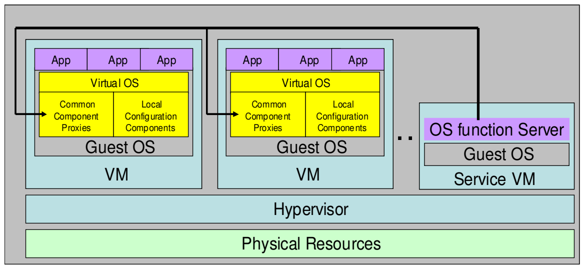

Figure 4. Dedicated VM with OS component server

The thin OS layer includes only the required VM and installed application and user specific components (such as drivers and configuration settings), and a set of proxies (pointing mechanism) for accessing the shared OS components. The VM instance hosting the OS server behaves as an independent VM instance, with the OS image stored locally. The OS server is highly scalable; instances of the VM with the OS server can be provisioned or removed in respond to workload. (The same holds true for using the file system approach as multiple copies of the OS images can be created on different disks or mounted on remote storage systems).

With the shared OS system, each individual VM image only needs to store the pointers (as part of the configuration parameters), its own applications and data instead of the entire OS copy; as a result, less storage capacity is required. Figure 5 shows the storage view of VM of OS servers and their client VM images. For large scale Cloud operation with thousands of VM images, the savings in storage cost is significant.

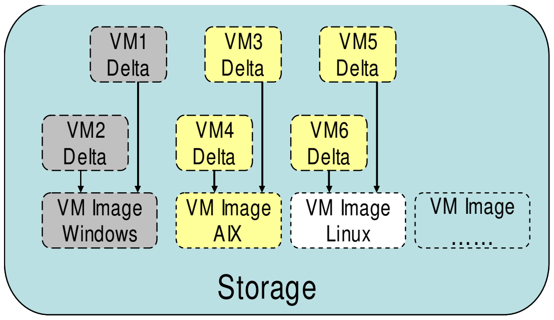

Figure 5. Storage view of master OS VMs

Another major potential benefit of the OS sharing system in Cloud is the significant time and reliability improvement in performing software patches and updates. The traditional approach of on-line software OS updates and patches add significant costs to the management of a Cloud environment, since each VM image needs to be powered up and shut down for the updates to be installed. In a control environment where all VM instances are centrally managed, such as data centers, the OS component sharing approach with the appropriate VM instance version management tools, a significant number of OS patches and updates can be performed only on the various OS component servers or file system with a single master image for each of target OS types. Subsequently, all changes will propagate to its entire set of VM instances upon re-start. The time and cost saving in performing regular and emergency updates and patches are substantial, in addition to the increased service reliability by reducing the risk associated with power up and down thousands of VM images while performing the traditional software updates and patches.

III. Linux Based Shared OS

There are numerous ways to design and implement the OS component sharing system, depending on the operating system type. For some OSs, it may not be possible without substantial changes to the OS structure (e.g. Windows), some OSs may have existing infrastructure that can be leveraged to achieve relatively simple design and easy implementation. For simplicity, we use the common Linux file mounting process as an example to show a possible way of how the OS component sharing among VM instances in a Cloud setting can be realized. In a typical Linux OS installation [13], **/boot** directory, which contains files used by the bootstrap loader, LILO and kernel images, along with some necessary modules which are needed during startup and system initialization, are kept and loaded locally. Others such as /lib/modules which contains the loadable kernel modules can be loaded on-demand (Figure 6).

```
|---root
|---boot
|---lib
    |-----modules
|----
```

Figure 6. Linux directory structure

File system mounting is a common process in Linux OS that sets up a file system for use by the OS simply by adding the mount information in the **/etc/fstab** configuration file [14]. Based on the mounting information entries in this file, file systems are automatically mounted when the system boots. Figure 7 shows an example of the file mount information added to **/etc/fstab** configuration file.

```
// network mount info in /etc/fstab file
9.7.25.123:/vol/shared_lib  /shared  nfs
ro,soft,bg,timeo=3,intr 0 0

// establish symbolic link
ln -s  /shared/lib/modules  /lib/modules
```

Figure 7. File mount and symbolic link example

A modified Hypervisor keeps a registry of the available OS component sharing servers. At VM instance start-up, the Hypervisor locates the **/etc/fstab** configuration file and insert the remote file mounting information to a Hypervisor selected OS component sharing server, making it available to the VM instance's OS after start-up. This change affects only the running VM instance, the original VM image remains unchanged. As far as the users and applications are concerned, all these changes are transparent to them and it looks to them as if they have the full OS and the isolated environment. This example takes advantages of the Linux

OS infrastructure and is the simplest, but not necessarily the best way to enable OS components sharing. Recently, there are numerous new ideas and developments in the Linux world (and other OSs) that attempt to provide a more flexible executing environment, which will provide a better foundation for the OS component sharing system. The recent advance of network boot [15] and diskless OS [16] is an example to move away from the traditional OS executing environment to meet the challenges of a new generation of applications and usages. Researchers may look into the Linux initial system boot process [17] and the kernel system for a better and more efficient design and implementation, but this is beyond the scope of this report.

IV. EXPERIMENT WITH VMWARE ESX

To prove the feasibility of the OS sharing architecture in a Cloud setting with a typical and widely used virtualization platform, we conducted a serious of experiments on a VMWare ESX 4 server [18] hosted on an IBM Blade server (IBM BladeCenter HS22 - 7870 - 4 GB RAM - 2.53 GHz) to simulate the operation of the OS sharing system. The idea is to create multiple snapshots from the original master image and support running independent VM instances from these snapshots.

A master VM image with complete OS, which serves as the repository or master copy for the common sharable OS components is created. We then utilized VMWare' Snapshot Manager [19,20] to provide the snapshot and instance creation functions. Although these functions can be performed using the Snapshot Management GUI, our entire process of snapshots and VM instances creation and the subsequent modification in the delta of the children images is done automatically by a script using VMWare's published APIs [21]. Figure 8 shows the high-level algorithm to create snapshots and independent VM instances from the master VM image.

In the example algorithm, the key idea is the use of VMWare ESX server's existing Snapshot Manager to create snapshots and VM instances are supported from these snapshots. Snapshots are utilized by many VMWare and third party products and features such as VMWare Consolidation Backup, VMWare Data Recovery, VMWare vCenter and the VMWare Infrastructure Client and VMWare Lab Manager. Obviously, the algorithm we developed for our experiment is not the intended use of snapshot mechanism by VMWare, but we were able to modify each of the VM instances' configuration parameters (e.g. new IP and host name as in step 6) in a way that each of the snapshot supported VM instances acts as independent VM instance from the perspective of the users (users can login in simultaneously into each of the created VM instances and communicate among them using their unique assigned IPs and host names). Internally from the Hypervisor, it acts as a typical VM instance created from the snapshot and managed by VMWare's snapshot management functions.

1. Create Master Image on VMWare ESX with fixed IP and embedded host public key (to allow remote login from script without password), this image is immutable
2. Create child image by cloning only the configuration files(.xmx, .vmdk, .vmxf) from the Master Image, assign new name to the new child image in configuration files and remain pointing to the Master Image's flat vmdk file
3. Register the child to the ESX Server
4. Create a snapshot instance from the child image's .vmx file (the cloned and modified configuration file in step 2)
5. Boot up and log on to child instance using the Master copy's IP, as it still uses the IP of the Master Image
6. Assign new IP and host name to the child instance, thus the new IP and host name are saved in the delta of the snapshot's image.
7. Reboot and login to new instance with the new IP established in step 6
8. Verify communication with the new instance by pinging the new instance using the new IP established
9. Repeat 2-8 for new instances

Figure 8. Algorithm of creating VM instances with a master VM image

Results:

Running the script produces a set of new snapshots and VM instances, all of these VM instances can be logged in simultaneously with their assigned IP addresses (step 6 of figure 8). Normally, it will take around 15 minutes to provision a VM instance from a VM image of roughly 10G in a typical server. With the automatic fast provisioning script, we were able to provision instances from the 10GB image without copying the entire master image in less than 1-2 minutes per VM instance, including the time to establish network communication, this is significantly less than the amount of time it takes for an instance provisioned in the traditional way from individual full images. The initial storage of the snapshot and delta image is relatively small, since it only includes the configuration files (<vm>.vmx, <vm>-<number>.vmdk, <vm>-<number>-delta.vmdk, <vm>Snapshot <number>.vmsn). The child delta.vmdk file which is initially created with the snapshot is a sparse disk. The virtual disk (delta) contains no data and no OS code in places, and yet is able to be used to provision independently running VM instance, which suggests that the OS code from the master image is used by the VM instances. As the VM instance runs and application data is added (via the COW – copy on write mechanism), it will grow in size up to the full storage size determined initially by the master image.

The purpose of this experiment is to prove indirectly the feasibility and performance of the shared OS approach in Cloud setting using a common and widely used Cloud platform (VMWare ESX server) and its supported utilities e.g. the Snapshot Manager, it is by no means the way or the only way to implement the shared OS system. We do not go into the internal memory paging mechanism for snapshot operation of the VMWare ESX Server platform and used only the available VMWare ESX APIs, as we believe that the implementation of the OS sharing idea in Cloud is platform specific. This experiment proves that the concept of shared OS in Cloud setting is possible even with the currently available and unmodified virtualization platform. In summary: we have demonstrated the concept of OS sharing by VMs in Cloud setting by a master VM image as the OS repository, and the created snapshot instances as the independent VM instances sharing the OS of the original master VM image. We have also showed that the size of the OS-sharing VM images are reduced, resulting in faster provisioning and reduced storage need.

V. Issues And Future works

As mentioned in earlier sections, in a large scale Cloud setting, dynamically sharing common components by VM instances is obviously beneficial in terms of image storage, ease of maintenance especially for software updates and patches, reliability and the more efficient use of resources. However, there are numerous issues to be explored before its full benefit and potential can be realized: (1) Not all current OS infrastructures can be adapted for component sharing. (2) This approach deviates from the current common Cloud model in which a VM image other than its own snapshots is completely isolated and self-contained. (3) A new Hypervisor may be needed for real-time instance modification which adds to complexity of the entire approach. (4) Dynamic component loading incurs system overhead which may affect performance especially when the common components swap in/out frequently due to custom applications. (5) Shared component may cause security concerns to some users. (6) Sharing OS components among VM instances represents a small step towards centralized computing, or at the minimum, a hybrid between total independence and centralization, this will cause controversies such as "Is some degree of centralized computing necessary to meet the current and future challenges of Cloud computing model ?". (7) Since running VM instances relies to some extent on external components, it becomes less portable. 8) As the size of the VM instance grows, the OS part of the image becomes relatively less important, and potential benefit of sharing OS in terms of storage and provisioning time decreases, but the potential benefit of more efficient OS patching and updates remain. These issues are very interesting topics, and present themselves uniquely in a Cloud setting. We also raised the question of the classical arguments of the advantages and disadvantages of centralized vs. distributed computing in a Cloud setting with service providers facing issues on economies of scale and management. Perhaps, a hybrid of centralized and distributed computing approaches will be a viable solution to address some of the Cloud computing issues, all these are interesting questions that may stimulate further research interests. In the future, we will focus on using the Linux OS, study and prototype tools to extract common components from various OSs, and different ways of efficient component sharing in the Cloud setting.

VI. Conclusion

In summary, our initial works have showed the concept and an approach to VM instance execution by enabling OS components sharing among VM instances in Cloud environment by sharing the OS of the master VM image. We have demonstrated indirectly this concept in a currently available and widely used commercial virtualization platform by creating and running multiple VM instances from a master VM image. We also explained that this approach benefits the Cloud computing service providers with increased storage efficiency, faster provisioning, efficient and reliable software updates and patches which are among the major pain points affecting customer satisfaction, efficiency and profitability of a Cloud service provider. We also identified some issues and obstacles with this approach and raised questions on how much centralization is needed to supplement the current Cloud computing management system to fully take advantages of the benefits of Cloud computing model.

References

[1] G. Gruman, "What cloud computing really means", InfoWorld, Jan. 2009.

[2] R. Buyya, Y. S. Chee, and V. Srikumar, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities", Department of Computer Science and Software Engineering, University of Melbourne, Australia, July 2008, pp. 9.

[3] D. Chappell, "A Short Introduction to Cloud Platforms", David Chappell & Associates, August 2008.

[4] Amazon elastic compute cloud (EC2). http://aws.amazon.com/ec2/.

[5] Enterprise Cloud: http://websphere.sys-con.com/node/1017378

[6] Jun Zhu, Zhefu Jiang, Zhen Xiao. Twinkle: A Fast Resource Provisioning Mechanism for Internet Services To appear in *Proc. of IEEE Infocom*, April 2011

[7] Virtual Machine Storage: http://www.gluster.com/solutions/use-case/virtualization/

[8] W. Zhou P. Ning, R Wang, Z Zhang, G Ammons and V. Bala, "Always Up-to-date – Scalable Offline Patching of VM Images in a Compute Cloud", ACSAC '10 Dec. 6-10, 2010, Austin, Texas USA

[9] Gautam Altekar, Ilya Bagrak, Paul Burstein, and Andrew Schultz.Opus: online patches and updates for security. In

SSYM'05:Proceedings of the 14th conference on USENIX Security Symposium,pages 19–19, Berkeley, CA, USA, 2005. USENIX association.

[10] A.C Amarie, T.V Dinh, G. Antoniu, "Efficient VM Storage for Clouds Based on the High-Throughput BlobSeer BLOB Management System", INRIA Sept 2010, 7434

[11] IBM DeveloperWorks, "Java programming dynamics, Part 1: Java classes and class loading", http://www.ibm.com/developerworks/java /library/j-dyn0429/

[12] Gesellschaft für Mathematik und Datenverarbeitung, "Progress in distributed operating systems and distributed systems management", European Workshop, Berlin, FRG, April 1989 Proceedings

[13] Linux Directory Structure, http://www.comptechdoc.org/os/linux/ /linux_ugfilestruct.html

[14] File Mounting, http://itc.virginia.edu/desktop/linux/mount.html

[15] Linux Network Boot, http://www.linuxtoday.com/infrastructure/ 2009051801935OSNT

[16] Linux Remote Booting a Diskless Computer: http://www.comptechdoc.org/os/linux/howtos/Howtoremoteboot//index.html

[17] IBM DeveloperWorks, "Inside the Linux boot process", http://www.ibm.com/developerworks/linux/library/l-linuxboot/

[18] VMWare URL: VMWare ESXi & ESX Information Center, http://www.vmware.com/products/vsphere/esxi-and-esx/index.html

[19] VMWare URL: Snapshot Manager Information, http://www.vmware.com/support/ws55/doc/ws_preserve_sshot_manager.html

[20] VMWare Knowledge Base: Working with Snapshots, http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1009402

[21] VMware Infrastructure (VI) API Reference Guide http://www.vmware.com/support/developer