International Journal of Computer Trends and Technology
ISSN: 2231 – 2803 / https://doi.org/10.14445/22312803/IJCTT-V69I10P106

Volume 69 Issue 10, 39-42, October 2021
© 2021 Seventh Sense Research Group®

*Review Article*

# Virtualization Technologies: Overview, Differences & Similarities

Omro Alawadh, Raif Abdulrahman, Fahad Alkhaldi, Khalid Alsheddi, Mohammad Buaysha, Humoud Rashidi

*Saudi Aramco, Area IT Department, Abqaiq, Saudi Arabia*

Received Date: 12 September 2021
Revised Date: 14 October 2021
Accepted Date: 27 October 2021

***Abstract*** *- Virtualization is a key technology in today's world. To use it effectively, one must understand how it works and identify which type best suits their needs. This paper defines virtualization and goes over its potential applications. The three main types of virtualization are also discussed in detail: OS-level virtualization, paravirtualization, and full virtualization. Finally, the similarities and differences among them are highlighted in terms of security, performance, features, and OS support.*

***Keywords*** *- Virtualization, Paravirtualization, Os-level, container, Performance, Cloud-computing.*

## I. INTRODUCTION

It is universally agreed upon that the more time people spend honing their skills, the more experience they gain in improving said skills. This was the case with the software industry, where a combination of people, hard work, and fierce competition led to great strides in software development and security. Virtualization, especially, has had one of the most noteworthy advancements and transformations since its early use. Not only has it provided security, but it also filled an important niche that changed our view of hardware. This paper aims to explore the different techniques developed for virtualization: OS-level virtualization, paravirtualization, and full virtualization while focusing on their similarities and differences.

## II. VIRTUALIZATION

Before we delve into the different types of virtualization technologies, let us first define virtualization and its importance. Virtualization is a technology that allows us to create virtual instances of computer services and resources, both physical and logical [5]. This technology is praised by security professionals for isolating between virtual resources and the physical host they reside on, making it harder for attackers that have compromised the virtual environment from reaching the host system. Perhaps the most significant breakthrough of virtualization technologies is allowing multiple isolated systems to exist on the same hardware, maximizing the cost-efficiency of servers [10]. This has allowed many startups and small and medium businesses to afford entry costs into the market [1]. It is also a key concept that drives cloud computing. Many businesses outsource their IT services to cloud providers, which is only possible due to virtualization. Indeed, it can be argued that without it, our world would not be the same. Therefore, it's important to analyze and compare the different types of virtualization and the solutions they offer. There are three types of virtualization: OS-level virtualization, paravirtualization, and full virtualization.

### *A. OS-Level Virtualization*

Also known as container virtualization. In this type of virtualization, multiple isolated instances are created within a system's kernel. These instances can be jails, containers, virtual partitions, virtual environments, or virtual private servers [4][6]. A system's kernel is the brain of the operating system that allows the interaction between applications and hardware [7]. All virtualization instances of this type share the same kernel as the host system. Therefore, creating a virtual window environment on a Linux machine won't be possible in OS-level virtualization. However, sharing the same kernel does come with its inherent advantages of having a smaller overhead, which translates to faster execution and better performance [15].

#### *a) Chroot & jails*

Starting with jail instances, "Chroot jail" is one of the earlier examples of OS-level virtualization being utilized by programmers. Chroot, short for change root, is a Unix command that allowed programmers to change their application's perspective of its directory, to behave as if it were installed in the root director jailed and unable to traverse upwards to the system directories. This affects all the applications and processes installed in a chrooted directory. Chroot was implemented by programmers for security reasons and was even considered a best practice for a long time in order to isolate attackers in the application's directory in the event the application was compromised. Unfortunately, this came with its own limitations, which required all libraries and APIs that the application was referencing to be in the same directory because it could not escape into the system directories to reach them. This meant programmers had to ensure all application dependencies were also copied under the chroot directory, which wasn't time or space-efficient.

Freebsd jail is another type of jail instance specifically for the Unix FreeBSD operating system. It implements a partitioning approach to achieve isolation. Each partition is referred to as a jail and is considered its own independent virtual environment, sharing the host system's kernel, as is the case with all OS-level virtualization instances[8]. Freebsd is an evolution of “chroot” that addresses its limitations by virtualizing the file system space. There are two types of Freebsd jails: “Complete” jails that virtualize entire Freebsd operating systems and “service” jails, which are used to isolate an application or a service [16].

***b) Virtual private servers (v-server)***

Abbreviated as Linux V-server, Itis a Linux software for creating multiple virtual private servers running concurrently on the same physical hardware. Each of the virtual private servers is an isolated Linux Operating System that can provide the full functionality of a Linux server. Linux V-server utilizes chroot, segmented routing, and other tools to achieve its isolation[17]

***c) Containers LXC& Docker***

Linux containers, abbreviated LXC, are container instances that allow users to create virtual environments with all the required dependencies to run any number of applications or services. Unlike chroot, LXC doesn't suffer from the same limitation where an application's libraries and APIs have to be copied to its directory. Containers typically use Cgroups& Namespaces to achieve their isolation. Cgroups allow managing & controlling of the system resources for a group of processes, mainly CPU memory utilization. The main benefit of Cgroups is to prevent a single process from using up all the system resources. On the other hand, namespaces obtain the required resources for an application to run and deliver them to the application. This allows isolation of the application's process from system resources or other processes [14].

Docker is a very popular type of container. It is an evolution of LXC that started out by using it but has long since moved away from it. Docker focuses on single application containment, unlike LXC, which can be used to run any number of applications[18]. This makes docker more lightweight. As a result, further boosts its portability [4].

***B. Paravirtualization***

As a step up from OS-level virtualization, paravirtualization allows for the virtualization of guest operating systems that don't necessarily share the same OS as the host machine. This is done by using a virtualization layer called a virtual machine manager (VMM) or hypervisor. In other words, the guest OS doesn't have to share the same kernel as the host. Hypervisors come in two types. Type 1 hypervisors, also called bare-metal hypervisors, run directly on top of the hardware, separating the OS from the hardware layer. Therefore, all the guest OSs interact directly with the hypervisor & the hardware rather than the host OS. On the other hand, a type 2 hypervisor, also known as a hosted hypervisor, runs within the host OS just like any normal application [11].

”Para” has the Greek meaning of ”with” or ”alongside”. This appropriately describes paravirtualization, as it is the type of virtualization where we virtualize a guest OS alongside the host [12]. This is achieved by running a hypervisor, VMM, or virtualization layer under the OS in ”Ring -1” [6]. The guest OS also knows it is being virtualized, as certain changes must be made to it to allow it to communicate effectively with the hypervisor [13]. Instead of directly executing privileged instructions, the guest OS coordinates the process with the hypervisor. In other words, system performance is improved by replacing privileged instructions that are hard to virtualize with easier ones that the hypervisor understands [10].

Paravirtualization's biggest flaw, however, is the need to modify the guest OS. This means closed source operating systems like Windows are not supported [6].

***C. Full Virtualization***

Like paravirtualization, full virtualization requires a hypervisor to manage the guest OSs[9]. Unlike it, however, the guest OS does not know it is being virtualized. This is due to the fact that the hypervisor presents the guest OS with a virtual interface that looks and acts like a normal physical computer. This means full virtualization supports unchanged operating systems, including Windows. Indeed, that very reason might be why it is the most popular form of virtualization used today [2].

However, to achieve that, there are high-performance penalties compared to OS-level virtualization or paravirtualization. The most basic form of full virtualization is hardware emulation. In this case, the hypervisor listens for instructions and then traps and executes any privileged ones. This process introduces latency and additional processing required for instruction translation. To mitigate this latency, some strategies are used to speed up or circumvent the translation process [13]. One of the strategies is called dynamic binary translation. It works by directly executing privileged instructions speeding up performance. In the event that a privileged instruction is required, the dynamic binary translator translates it into an unprivileged executable instruction.

***a) Hardware-assisted virtualization***

Despite the speedup from dynamic binary translation, there is still a noticeable performance penalty on full virtualization [9]. As a solution to this problem, processor manufacturers have implemented virtualization support on their CPUs, namely Intel-VT and AMD-V. Both of these implementations work by introducing another privilege level to the CPU known as “Ring -1,” similar to paravirtualization. After which, additional instructions are included in the architecture that is specialized for virtual machines. The hypervisor running the guest OSs must support hardware assistance and the use of special instructions. The idea here is that guest OSs run at a higher ringing which privileged instructions are directly executed through the use of special instructions [3].

### *D. Comparison*

Even though all three aforementioned virtualization techniques lie under the virtualization umbrella, each comes with its own distinctions.

#### *a) Security*

All three types of virtualization utilize the idea of separation or encapsulation, where the virtual instance is separated from the outer system. However, not all three share the same degree of separation. In testing, several OS-Level virtualization systems showed poor isolation. In fact, only the CPU could be properly isolated from the rest of the system [14]. Paravirtualization has a similar problem as well, in which all guest OSs share the same kernel and are aware they are being virtualized. Conversely, Full virtualization is the most secure as each virtual machine runs separately and independent of the other, providing overall the best isolation [10].

#### *b) Performance*

Despite the security advantage of full virtualization, it lacks in terms of performance. While full virtualization experiences a loss of performance, as previously discussed, requiring hardware assistance and other workarounds. Container or OS-level virtualization can achieve near-native performance [14]. Paravirtualization also shows great performance figures in contrast to full virtualization as guest Oss know they are being virtualized and can communicate with the hypervisor.

#### *c) Features*

Another aspect of comparison is the feature set each type of virtualization includes. Full virtualization and paravirtualization systems allow features such as full migration, suspension and resumption, checkpoints, and hardware independence. On the other hand, OS-level virtualization requires support from kernel developers to implement such features [14].

#### *d) OS Support*

The last important distinction to make is in guest OS support. OS-level virtualization comes last in this regard due to its inability to host a different OS version than the host[13]. By contrast, Paravirtualization can run multiple different OSs as long as they are modified to support it. Finally, full virtualization provides the best OS support as it requires neither modification of host nor guest OS [1].

## III. SUMMARY

In summary, virtualization as a whole is a very useful concept. It allows us to better utilize existing infrastructure as well as reduce the cost of building new infrastructure. There are many types and techniques used for virtualization as they all fall under one of the three categories: OS-level virtualization, paravirtualization, and full virtualization. Each of the three types comes with its own advantages and disadvantages. Overall, OS-level virtualization provides the best performance but has limited applications. Full virtualization has the most potential applications and support but also suffers from the highest performance penalty. Paravirtualization tries to strike a balance between the two but has its own setbacks in terms of security and OS support.

## REFERENCES

[1] GhannamAljabari., Virtualization of IT infrastructure for small and medium businesses, In: 2012 International Conference on Communications and Information Technology (ICCIT), International Conference on Communications and Information Technology (ICCIT), (2012). June 2012, pp. 129–133. doi: 10.1109/ICCITechnol.2012.6285775.

[2] Adriano Carvalho et al., Full virtualization on low-end hardware: A case study, In: IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society. IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society., (2016) 4784–4789. doi: 10.1109/IECON.2016.7794064.

[3] Wei Chen et al., A Novel Hardware-Assisted Full Virtualization Technique . In: 2008 The 9th International Conference for Young Computer Scientists. 2008 The 9th International Conference for Young Computer Scientists, (2008) 1292–1297. doi: 10.1109/ ICYCS.2008.218.

[4] Scott Hogg. Software Containers: Used More Frequently than Most Realize. Network World. May 26, (2014), url: https://www.networkworld.com/article/2226996/softwarecontainers--used-more-frequently-than-most-realize.html (visited on 09/20/2020).

[5] Nancy Jain and Sakshi Choudhary., Overview of virtualization in cloud computing, In: 2016 Symposium on Colossal Data Analysis and Networking (CDAN). Symposium on Colossal Data Analysis and Networking (CDAN),(2016) 1–4. doi: 10.1109/CDAN.2016.7570950.

[6] Barrett, D. and Kipper, G., (2010). How Virtualization Happens. [online] ScienceDirect. Available at: <https://www.sciencedirect.com/topics/computer-science/level-virtualization> [Accessed 20 September 2020].

[7] Madhavan Nagarajan., An Overview of Operating Systems and Explanation of the Kernel. Medium. July 30, (2019). url:https://levelup.gitconnected.com/operating-systemand-kernel-ef76f4d0bd8e (visited on 09/20/2020).

[8] Rani Osnat., A Brief History of Containers: From the 1970s Till Now. Jan. 10, (2020). url:https://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chrootto-docker-2016 (visited on 09/20/2020).

[9] A. B. S. et al., System Performance Evaluation of Para Virtualization, Container Virtualization, and Full Virtualization Using Xen, OpenVZ, and XenServer, In: 2014 Fourth International Conference on Advances in Computing and Communications, (2014) 247– 250.

[10] Amir Ali Semnanian et al., Virtualization Technology and its Impact on Computer Hardware Architecture, In: 2011 Eighth International Conference on Information Technology: New Generations, 2011 Eighth International Conference on Information Technology: New Generations. (2011) 719–724. doi: 10.1109/ITNG.2011.127.

[11] Michael Terrell and Natarajan Meghanathan., Setting Up of a Cloud Cyber Infrastructure Using Xen Hypervisor, In: 10th International Conference on Information Technology: New Generations. 2013 10th International Conference on Information Technology: New Generations, (2013) 648–652. doi: 10.1109/ITNG.2013.100.

[12] Understanding Full Virtualization, Paravirtualization, and Hardware Assist. Sept. 11,(2007)., url: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/VMware_paravirtualization.pdf (visited on 09/22/2020).

[13] John Paul Walters et al., A Comparison of Virtualization Technologies for HPC, In: 22nd International Conference on Advanced Information Networking and Applications (aina 2008). 22nd International Conference on Advanced Information Networking and Applications (aina 2008). ISSN: 2332-5658, (2008) 861–868. doi: 10.1109/AINA.2008.45.

[14] Miguel G. Xavier et al., Performance Evaluation of Container-Based Virtualization for High-Performance Computing Environments, In: 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. ISSN:2377-5750. Feb. (2013) 233–240. doi: 10.1109/PDP.2013.41.

[15] Yang Yu., OS-level Virtualization and Its Applications, Ph.D. thesis. Dec. (2007). url:https://dspace.sunyconnect.suny.edu/bitstream/handle/1951/448 96/000000243. sbu.pdf?sequence=3 (visited on 09/22/2020).

[16] Riondato, M., (2020). Chapter 14 Jails. [online] Freebsd.org. Available at: <https://www.freebsd.org/doc/handbook/jails.html> [Accessed 29 September 2020].

[17] Linux-vserver.org. n.d. Overview - Linux-Vserver. [online] Available at: <http://linux-vserver.org/Overview> [Accessed 29 September 2020].

[18] Banerjee, T., (2014). LXC Vs. Docker. [online] Archives.flockport.com. Available at: <https://archives.flockport.com/lxc-vs-docker/> [Accessed 29 September 2020].