A Survey of Virtualization, Virtual Machines, and Containers

Eugene Triguba

# Abstract

This paper examines virtualization, virtual machines, and containers. It attempts to clarify what they are, how they are used, and what the advantages and disadvantages of each are. Furthermore, this paper goes on to two case studies where VMware and Docker are examined.

* 1. Index Terms: Virtualization, Containers, Virtual Machines, Docker, VMware

# I. Introduction

Virtualization and the various methods of utilizing it are popular right now, especially for use cases in the cloud. However, it is far from a new concept. It’s a concept that has been around since the 1960s [Tanenbaum and Bos 2014]. In 1974, a seminal paper titled “Formal Requirements for Virtualizable Third Generation Architectures” was published by Gerald Popek and Robert Goldberg which laid out the precise conditions a computer architecture should follow in order to support virtualization in an efficient manner [Tanenbaum and Bos 2014]. Despite this relatively long history, virtualization had not gained wide spread adoption until around 1999 when Stanford researchers developed a new hypervisor called Disco, which later when on to become the virtualization company VMware [Tanenbaum and Bos 2014]. Further, Intel did not directly support virtualization for its x86 line of processors until 2005 [Tanenbaum and Bos 2014]. The resurgence of an interest in virtualization is partly due to computers now having enough processing power to use virtualization to partition resources on a single server into multiple execution environments [Medina and García 2014]. These units are termed Virtual Machines or VMs.

The two implementations of virtualization that we will be exploring in this survey are virtualization with virtual machines and with containers. In part II of this paper, the research on virtualization, virtual machines, and containers will be covered. We’ll do this by surveying current research to look for the definitions of each concept, the history, and the advantages and disadvantages of each approach. This will give us the background knowledge of the broader general concepts to explore the more specific case studies in part III. In these case studies, we’ll look at virtualization implemented with virtual machines and containers, namely, VMware and Docker. VMware was chosen because it was a front-runner with bringing virtual machines forward to mainstream use and is still immensely popular to this day. Docker was chosen to explore further because it is by far the predominant implementation of a container-based solution.

# II. Background Research

In these background research sections, we discuss virtualization, virtual machines, and containers.

## A. Virtualization

First off, what is virtualization? Virtualization is a technology that allows you to create a virtual resource. In more practical terms, say you have a number of physical servers for various services, such as a web server, hosting legacy applications, etc. However, all these servers are only using a small fraction of each their capacity. With virtualization, you can split a single physical server into separate virtual ones. That means if you had one physical server for a web application that was taking up 20% of that server’s capacity and another physical server for some legacy applications that were taking up 30% of the second physical server’s capacity, you could consolidate them into one physical server with virtualization, thus saving money and utilizing the hardware more efficiently.

Hypervisors, also called virtual machine monitors, were one of the first technologies to enable virtualization. Reverting back to Popek and Goldberg’s paper, there were three requirements that a hypervisor must do in order do virtualization efficiently. Indeed, the first is the efficiency property which says that all instructions that are not potentially harmful should be directly executed by the hardware without any intervention by the hypervisor [Popek and Goldberg 1974]. The second is called the resource control property which simply states that the hypervisor should be in full control of its virtualized resources; the program running in the virtualized environment should not be able to affect the system resources [Popek and Goldberg 1974]. Lastly, there is the equivalence property which states that the behavior of the program inside the virtual machine should be the same as the behavior outside the virtual machine on the bare hardware [Popek and Goldberg 1974]. Those are the three requirements for an efficient virtualized environment. The trick with virtualization is to trick the guest operating system that is on top of the hypervisor to think it is running on real hardware.

One of the reasons to use virtualization is the security benefits from doing so. However, Fred Cohen in “The Virtualization Solution” [2010] argues that it might not be the saving grace and lays out a blunter image of virtualization. Cohen chronicles us through various pieces of technology that were thought of as being the solutions to computing security: from operating systems and concepts of separation, to small security kernels, to perimeter controls, to intrusion detection, to encryption, and to now virtualization. He then goes on to state that none of these technologies will save us and attempts to detail what we can expect and what we shouldn't expect with virtualization. So, what can we expect? Cohen says virtualization is a form of separation, a separation of one virtual thing from another, and that virtualization will only be as good as the mechanism that will be implementing it. He goes on to say that we haven't been able to build a kernel or an OS that meets our security needs, so why would virtualization any different? However, virtualization is providing at levels where you typically don't see it being applied and is being used to be able to disaggregate risks [Cohen 2015].

One of the main advantages of employing virtualization technologies is in cloud computing. The cloud is simply offloading your computing and storage needs to a data center. Virtualization is heavily used in the cloud. The isolation provided by VMs allows multiple clients to share a single physical machine. They may also choose to use containers, which use less resources and are more lightweight. Therefore, they are also more scalable. Virtualization allows the cloud vendor to achieve lower server maintenance costs by improving the utilization of any one server, thereby also reducing power consumption [Lee et al. 2019].

The virtualization technologies can be briefly categorized into two types: the hypervisor-based full virtualization and the container. The hypervisor-based full virtualization creates virtual machines on top of a hypervisor, which is a software layer between the host computer and the virtual machine [Lee et al. 2019]. The next one utilizes operating-system-level virtualization. This type of virtualization divides up a single running instance of a operating system into multiple partitions, called containers [Lee et al. 2019]. The hypervisor-based solution utilized a hardware-level virtualization. The difference between hardware and operating system virtualization is that in hardware-level virtualization the guest OS thinks it’s running with its own dedicated hardware. In operating-system-level virtualization, the application thinks it’s running its own dedicated OS. In both cases, these assumptions are false since all the virtual machines are sharing the same hardware and all the containers are sharing the same OS, but the illusion is important.

## B. Virtual Machines

The architecture of IBM’s M44/44X was the first computer to introduce the term virtual machine [Medina and García 2014]. A virtual machine emulates a computer system. There are multiple kinds of virtual machines. There are system virtual machines, also called full virtualization virtual machines. These VMs substitute for a real machine, providing the functionality you would need to run an entire operating system. There are also process virtual machines. An example of this would be the software called WINE. Process virtual machines execute a computer program in an environment that is independent of a particular platform. There are also different kinds of hypervisors that run virtual machines: type 1 and type 2. A type 1 hypervisor is one that runs on the bare metal. There is no operating system it needs to communicate with, thus making it more performant. Type 2 hypervisors are more common in general use and are ones that run on top of an operating system. They are forced to go through the hypervisor and the operating system to execute instructions, making them slower. However, hypervisors nowadays will try to be out of the way for the majority of instructions, as long as they don't pose a threat.

The advantages to using virtual machines comes partly from the strong isolation they provide. Since all your binaries/libraries, applications, operating systems, etc. are all copied and run on top of your hypervisor for each running instance of a virtual machine, this provides strong isolation between any given virtual machine. Furthermore, virtual machines allow higher utilization of any particular machine which in the end can save money on hardware, electricity, and rack space. VMs are quite useful when it comes to legacy applications, allowing you to run operating systems that are no longer supported or no longer runs on current hardware. Another advantage is would be in software development. With the advent of virtual machines, multiple physical computers no longer need to be set up just to test an application. This reduces the setup time from possibly days to minutes.

## C. Containers

Container virtualization is often referred to as a “lightweight virtual machine.” Each container shares the operating systems’ host kernel, thus giving the container similar performance to that of non-virtualized systems [Lee et al. 2019]. It also shares resources that other containers may need as well, reducing the need to duplicate them in each container. In 2014, IBM released research that suggested the average container is 26 times faster than a VM on a per-transaction basis [Anderson 2015]. This performance combined with the smaller storage footprint combine to allow you to pack a server with many more containers than you would be able to with virtual machines. Another important aspect is startup time. Containers spin up and spin down on the order of milliseconds. This isn't the case for virtual machines, which often require several minutes. However, they lose some level of isolation given that each container doesn’t use its own individual operating system. Containers utilize process-level isolation rather than being fully isolated like virtual machines, which is considered less secure. This aspect is one of the main barriers to adoption.

The discussion of containers would not be complete without discussing microservices. The microservice architecture is closely related to the advent of containers. This is an application where containers shine. Microservice architectures help build applications by consisting of loosely coupled parts that are able to operate independently [Sultan et al. 2019]. Before microservice architectures, the vast majority of applications used to be monolithic, which is an application whose parts are tightly coupled and can't be executed separately [Sultan et al. 2019].

# III. Case Studies:

In these case studies, we’ll take a look at two technologies that build upon the idea of virtual machines and containers: VMware and Docker.

## A. VMware

VMware, as stated in the introduction, was first developed by three Stanford researchers in 1997, although originally called Disco in its prototype version [Tanenbaum and Bos 2014]. VMware was founded in 1998 with the goal of bringing virtualization support to the x86 architecture as well as to personal computers [Tanenbaum and Bos 2014]. They were largely the front-runner in this domain. However, despite VMware being the leading commercial provider of virtualization products, there are many other alternatives such as VirtualBox (open-source and free, all platforms), KVM (built into Linux), and Parallels (Mac).

VMware makes both type 1 and type 2 hypervisors, namely, ESX Server and VMware Workstation. ESX servers runs directly on the bare hardware whereas VMware Workstation runs on top of an operating system, with cross platform support for Windows, Mac, and Linux. VMware also makes a product they called vSphere which is used to manage clusters of servers that are running virtual machines. They even later came out with VMotion as well, allowing for the first time an administrator to move a running computer from one location to another without restarting or rebooting anything [Tanenbaum and Bos 2014]. In a paper titled "Performance examination of type-2 hypervisors: case of particular database application in a virtual environment", Timcenko et al. [2014] found that VMware provided the best performance for all cases except those when working with large files; Oracle's VirtualBox is preferred in that instance. Microsoft Virtual PC came out with the worst performance in their study.

There were many challenges with attempting to bring virtualization to the x86 architecture. One of the first challenges was that the x86 architecture was not even virtualizable; it didn't pass Popek and Goldberg's criteria [Tanenbaum and Bos 2014]. Another challenge stems from the complexity of the x86 architecture, which stems from the decades of backward compatibility the line needed to support. However, there were a couple key insights that allowed VMware to be able to virtualize the processes: that the trap-and-emulate direct execution could still be used some of the time and that by properly configuring the hardware, they could use dynamic binary translation at near-native speeds [Tanenbaum and Bos 2014].

## B. Docker

1. Docker is a container virtualization technology. It is a piece of software that helps you build, run, and manage containers. An interesting thing is that you can even run Docker containers within virtual machines [Docker 2019]. Docker also provides what they call a “developer workflow,” which is about being able to easily package up your environment and share it among your teammates [Anderson 2015]. The main advantage of using Docker over some sort of virtual machine is speed and performance. With virtual machines being a fairly large compute resource, Docker, containers in general, provides a more lightweight and performant computing resource. The technology came out of the advent of the DevOps software development practice and the microservices architecture where both raised a need for a faster solution than virtual machines [Anderson 2015].
2. The core of Docker is what is known as the Docker engine. The engine consists of a server, a REST API, and the command line interface. The server is a long-running program referred to as a daemon process which is what manages all the Docker objects, such as the networks, images, containers, and data volumes [Docker 2019]. On the client side, we have a command line interface to be able to talk to the server. In between those two layers, we have the REST API. The REST API is the interface between the server and the CLI that specifies how to talk to the server and what you can instruct it to do [Bui 2019].
3. Docker also has something called a registry. Docker Hub is docker's own public registry. However, there are others, and you could even create your own private registry if you so desired.
4. The registry is what stores Docker images. An image is a template that contains instructions on creating a Docker container, and images are often based upon other images [Docker 2019]. You could build an image that is based on Ubuntu but then also install PostgreSQL or MySQL for your database and an Apache web server, as well as your application. If you needed to, you can create your own image by writing a Dockerfile. Each instruction that is in a Dockerfile creates a single layer in an image; when you change the Dockerfile and rebuild the image, only what has changed is rebuilt [Docker 2019].
5. Docker used to use Linux's LXC to create Docker containers. However, since version 0.9, Docker now uses lib-container, their own virtualization format. One of the interesting things about Docker is how it is able to implement process isolation. Since Docker is implemented utilizing operating system virtualization, process isolation is one of the key features for isolating and protecting containers from one another. It turns out, Docker heavily relies on two Linux features to accomplish this task: namespaces and cgroups [Anderson 2015]. The namespace is what determines what your process has access to. When you run a new process on the Linux kernel, you're making a system call to a namespace, and the kernel will assign you to a namespace that has that process and any other resources you had requested [Anderson 2015]. Currently, Docker has five namespaces it provides to each container: mount, hostname, inter-process communication, process identifiers, and network [Bui 2019]. Controls groups, cgroups, are what allow you to control and manage the resources the individual container already has. It allows you to add and drop capabilities as you need it which gives you a granular level of control [Anderson 2015]. These are the two capabilities that allow Docker to safely create virtual environments for their containers.
6. Even though security is often the issues that is touted as the barrier to adoption with containers, Thanh Bui in an "Analysis of Docker Security" [2019] concluded that Docker is fairly secure, even with the default configuration. He goes on to say that the security level could be increased though if the Docker containers are run in a non-privileged mode and enables various solutions to harden the containers such as AppArmor or SELinux. AppArmour and SELinux are both security enhancements to the Linux system. Google has also recently come out with a product called gVisor which is a sandbox runtime that attempts to provide a virtualized container environment. gVisor specifically integrates with Docker, containerd (a container runtime), and Kubernetes (software that automates the deployment, scaling, and management of containerd apps).

# IV. Further Research

One of the things this paper is missing is a further discussion of the cloud. It was not the topic of this paper, but virtualization is used heavily in that domain and is in fact one of the main use cases of virtualization. That would be the most glaring omission for the treatment of this subject.

# V. Conclusion

From this paper, we've learned what virtualization, virtual machines, and containers are. Virtualization is a concept where we can create virtual resources. This has many benefits in the forms of higher utilization of computers which in turn can save on costs, quicker development time when you no longer need a second physical machine, and security benefits. We learned that virtual machines provide the strongest isolation and are often used with legacy applications but that they are a heavyweight computing resource. Furthermore, we learned that containers provide a more lightweight option and are heavily used with microservice architectures.

References

[1] A. Tanenbaum and H. Bos, Modern Operating Systems, 4th ed. Pearson, 2014, pp. 471-516.

[2] F. Cohen, "The Virtualization Solution", IEEE Security & Privacy Magazine, vol. 8, no. 3, pp. 60-63, 2010. Available: 10.1109/msp.2010.108.

[3] M. Pearce, S. Zeadally and R. Hunt, "Virtualization: Issues, security threats, and solutions", ACM Computing Surveys, vol. 45, no. 2, pp. 1-39, 2013. Available: 10.1145/2431211.2431216.

[4] S. Vaughan-Nichols, "Virtualization Sparks Security Concerns" in Computer, vol. 41, no. 08, pp. 13-15, 2008. Available: 10.1109/MC.2008.276

[5] C. Anderson, "Docker [Software engineering]" in IEEE Software, vol. 32, no. 03, pp. 102-c3, 2015. Available: 10.1109/MS.2015.62

[6] T. Combe, A. Martin and R. Di Pietro, "To Docker or Not to Docker: A Security Perspective" in IEEE Cloud Computing, vol. 3, no. 05, pp. 54-62, 2016. Available: 10.1109/MCC.2016.100

[7] V. Medina and J. García, "A survey of migration mechanisms of virtual machines", ACM Computing Surveys, vol. 46, no. 3, pp. 1-33, 2014. Available: 10.1145/2492705.

[8] Docker, 2019. *Docker Documentation*. [online] Docker Documentation. Available at: https://docs.docker.com/ [Accessed 4 May 2019]. ­­

[9] Gerald J. Popek and Robert P. Goldberg. 1974. Formal requirements for virtualizable third generation architectures. Commun. ACM 17, 7 (July 1974), 412-421. DOI: http://dx.doi.org/10.1145/361011.361073

[10] Lee, M., Park, S., Song, Y. and Eom, Y., 2019. Introspection of Virtual Machine Memory Resource in the Virtualized Systems. *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*.

[11] Sultan, S., Ahmad, I. and Dimitriou, T., 2019. Container Security: Issues, Challenges, and the Road Ahead. *IEEE Access*, 7, pp.52976-52996.

[12] Valentina Timcenko, Borislav Djordjevic, Slavica Bostjancic Rakas, and Nikola Davidovic. 2014. Performance examination of type-2 hypervisors: case of particular database application in a virtual environment. In *Proceedings of the International Conference on Information Systems and Design of Communication* (ISDOC '14). ACM, New York, NY, USA, 122-126. DOI: http://dx.doi.org/10.1145/2618168.2618187

[13] Bui, T., 2019. Analysis of Docker Security. *arXiv.org*.