The VM Hypervisor

William Roberts
r16n368
Montana State University
Bozeman, Montana

Nicholas Hager
d26w834
Montana State University
Bozeman, Montana

Janet Madrid
v11c372
Montana State University
Bozeman, Montana

Logan Ladd
c93j984
Montana State University
Bozeman, Montana

ACM Reference Format:
William Roberts, Nicholas Hager, Janet Madrid, and Logan Ladd. 2020. The VM Hypervisor. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

This paper will produce an understanding of hypervisors. We will look at the history of hypervisors, use cases for them, and how they interact with the host machine's resources. The focus of the paper and for our analysis we will discuss the differences in type 1 vs. type 2 hypervisors and why a user would prefer one over the other. With an ending discussion of virtualization today and future analyses or improvements.

Starting with a high level description of a hypervisor, A hypervisor is a piece of software that runs above a physical host machine and enables multiple virtual machines(VMs) to run alongside each other. A hypervisor is also known as a virtual machine monitor (VMM) because it separates virtual machines from each other by assigning each virtual machine its own piece of the underlying computing power, memory, and storage. As the name VMM implies, hypervisors prevent the virtual machines from interfering with each other.

2 BACKGROUND

2.1 History

The Hypervisor surfaced in the 1960's under the umbrella of z/VM, IBM's at-the-time ground-breaking virtualization software for the System Z computer. In IBM's research and development lab known as the Cambridge Scientific Center, the original Hypervisor known simply as Control Program, would come about to create multiple independent Virtual Machines that echo the underlying hardware. This Control Program or Cambridge Monitor System would develop into the 1970's with multiple versions, such as the CP-40, CP-67, etc. [3]

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Moving into the 1970s, the CP/CMS software would be made available to the general public as the VM/370 in 1972. This was notably done at the same time with the first release of servers with virtual memory, the S/370.

IBM continued to consistently supply source code and a community surrounding VM's quickly came to fruition. The functionality of the VM to run many operating systems at the same time while providing individuality from each other amidst running on the same hardware was the key to acceptance into the computing world.

The 1980s brought enhancements and changes to the z/VM software. Many users started using the software for development and deployment. A large contribution came from Olivia Newton John who implemented two levels of virtualization, allowing users to share hardware in a way where performance is shared across Operating systems, but spikes in processing could be re-allocated if need be. It was also in this time that the Virtual Machine Family split into three main products: the VM/SP, HPO, and VM/XA, each for different applications of speed and volumes of servers.

The 1990s and 2000s was largely a revolution for convergence of many VM software systems including hypervisors into a singular product called VM/ESA. This catered to many Linux applications as well as business applications. Later the 64-bit version of the z/VM architecture would enter the market in 2001, with 12 releases up to 2012. It was at this time that VM's were firmly solidified into the world of computing, and market interest placed virtualization into a mainstay of the economy. [3]

2.2 Importance

Hypervisors have many important applications in computing and the world of virtualization, the most important being how they influence server management. Hypervisors simplify server administration because in-line with other virtual machines, they are independent of the host environment. This means that the operation of a singular virtual machine does not affect other virtual machines or the underlying hardware. Even if one virtual machine crashes, others can continue operating without affected performance. Although this is true, it is important to note that this is not the case if there is a hardware issue. Hypervisors can therefore provide an environment where many users can work off the same hardware independent of each other, similar to if they were all on different computers. Advantages come to server administrators that can manage each VM as an inter-connected network. This is significant and is used in education, business, and home applications all over the world.

Figure 1: The VM/370, a Consumer VM

Hypervisors also have minor practicalities that make them an even more effective tool for server administrators. Administrators are enabled to move virtual machines in-between servers. This is a useful capability as it allows workload balancing if need be. Another small use-case is Hypervisors can be used to quickly test programs across different operating systems, For example, the server administrator could deploy a program into a shared network folder for the virtual machines, and each user on a different virtual system could quickly run and test the program.

Another important use case for hypervisors is the ability to consolidate servers in larger cloud environments. Hypervisors are exemplary in solving the problems that come with underutilized servers. Instead of running different volumes of workloads on separate machines or physical hardware, virtualization allows system administrators to consolidate and leverage the unused hardware to process multiple workloads simultaneously, creating a dynamic computing environment where time and work is saved due to full utilization of hardware.

Figure 2: VM Practicality

2.3 Research Motivation

As most of society would likely come short if asked what a hypervisor is, it is important for students, users and consumers to understand the underlying structure of their everyday computers and other tools. Many individuals may come to a desk job everyday for forty years and work at their monitor, keyboard, and mouse never knowing that there is a multi-thousand dollar machine sitting in a closet somewhere running all the machines in the company as a whole. This is achieved again, through hypervisors. The spread of knowledge is important surrounding this subject, because in order for businesses and other consumers to keep advancing in time consumption, overhead cost, and efficiency through its software, they must understand the tools they are working with. Many students also may find an interest in the field of virtualization or hypervisors through research such as this, and pursue a career to better society by improving on those three important aspects of business, education, or consumer use as previously described.

This research aims to be significant by providing an educational view of hypervisors, their history, their importance, a research summary of VM/host interaction, use cases for different types of hypervisors, an overview of the two primary types of hypervisors with an analysis of each, modern advancement of hypervisors that makes research such as this even more important as technological advancements are made ever so often in the 21st century, a discussion of the research, and conclusion.

2.4 Objectives

As briefly outlined in the former section, the goals of this research document are the following:

(1) To provide an overview of hypervisors that is relatively easy for any reader to understand.
(2) To educate the reader in a brief history of virtualization so that a context of how new virtualization actually is established.
(3) To educate the reader in the importance of hypervisors so that is it understood that hypervisors are all around in business, education, and consumer applications. It should also be understood that the primary goal of virtualization is efficiency in almost any context.
(4) To research how a hypervisor interacts with a host, the host being either hardware or an operating system depending on the type of hypervisor. The research is to be conducted by aggregating information from many different sources, including direct sources such as IBM that invented many of the technologies themselves.
(5) To research more in-depth the use cases of hypervisors, whether it be system administration, workload leveraging, heavy processing tasks among multiple machines, testing and quality assurance across different operating systems, and other use cases.
(6) To research and provide an in-depth view of Bare Metal Hypervisors.
(7) To research and provide an in-depth view of Operating System hosted Hypervisors.
(8) To provide an analysis of both types of hypervisors, and compare and contrast each.

(9) To research and provide an overview of modern advancements in hypervisors, as we live in a society where computing is extremely new relative to world history, and new technology surfaces every day.
(10) To provide a brief discussion on hypervisors from a student researcher standpoint to provide some less professional insight on the subject.

3 METHODS AND RESULTS

3.1 Type 1 (Bare Metal) Hypervisors

There are two distinct categories of hypervisors used today, both of which have differing characteristics and purposes. In this section, we will go over Type 1 Hypervisors, sometimes referred to as Bare-metal hypervisors because of how they run directly on the underlying computer's physical hardware.

Bare metal hypervisors were the first of the two types to be invented, going into development at IBM's Cambridge Scientific Center in 1960 [3]. The focus of this software was to enable more efficient and stable server infrastructure in the form of virtualization. Back then, networks had an issue where a bug in one user's system could cause another user's system to also crash, so it was key to be able to isolate virtual machines from one another. This idea led to the development of Full Virtualization, or *native virtualization*, where a virtual machine manager mediates between the guest operating systems and the native hardware [11].

This is the one important aspect of Bare-metal hypervisors, they are *mediators*. Because the VMM sits directly on top of the underlying hardware, it needs to mediate between the guest operating systems and the hardware itself (see figure: 3). Each of the guest operating systems (OS) are going to assume that they have full access to machine hardware like the processor, memory, etc., so certain OS related instructions need to be trapped and handled within the hypervisor. In this case, the Hypervisor takes the place of a "host" OS, as it handles the interrupts and other kernel-level calls of the guest OSes.

Usually alongside the OSes there needs to be some sort of management device that directs the VMM's process itself. This separate management machine can administer the different VMs as well as control the host hardware, making sure that everything is running as ideally as possible. Usually these management machines will conduct tasks like load-balancing between the VMs, VM sprawl prevention (cleaning up disused VMs), and VMM auditing. Overall the focus of the management device is to make sure that the user has a set of comprehensive management tools that support the hypervisor.

Because type 1 hypervisors run directly on physical hardware, they are usually incredibly efficient. This makes them ideal for computation heavy processes like running many virtual servers at the same time, providing virtual desktop integration (VDI) to entire teams of employees, and other practical things. Due to their efficiency they have gained a lot of traction in enterprise level development, being used mainly as powerful devices for server handling and development. Today, Type 1 virtual machines are competing with containerization for the future of Cloud infrastructure [9].

There are several popular Type 1 hypervisors in the market today, the most used of which are VMware vSphere/ESXi, Microsoft Hyper V, and KVM (Kernel-based Virtual Machine). VMware is inarguably the market leader, but each of these different hypvervisor technologies are powerful in their own right; each focusing on different commercial editions that cater to different enterprise needs. Most of them have a free limited edition as well, but KVM is free and open source, attributing to its popularity.

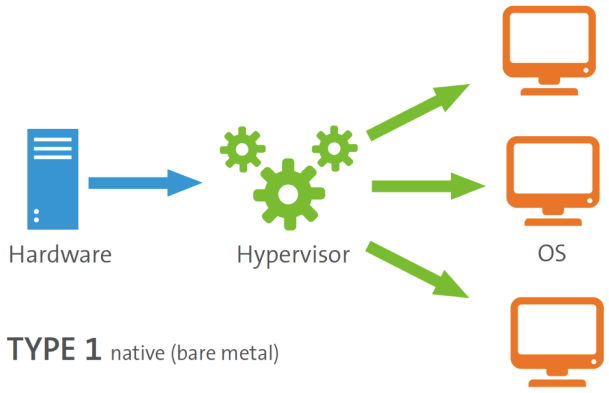

Figure 3: A Bare metal Setup

We can see now the power of Type 1 Hypervisors, especially within enterprises, but what use do they have for us, the lowly computer science student? Next, we will discuss the other category of Hypervisor, the Hosted VMM, a much more personal tool.

3.2 Type 2 (OS/Hosted) Hypervisors

Virtualization is a very powerful idea, enabling the use of many operating systems concurrently on a single machine completely changed the entire software industry and shaped how servers were developed. Turns out, running multiple OSes at the same time is incredibly efficient, and can also be applied to more than just server infrastructure. This is where the second type of Hypervisor comes in; the Operating System or Hosted Hypervisor.

This category of virtual machine manager focuses on bringing virtualization to the end-user, focusing on developer productivity [11]. Because the focus is on the end-user, these hypervisors forego running "bare metal" and instead operate as an application within an OS, much like any other process a user might run (see figure: 4). These hypervisors enable "quick and easy access to an alternative guest OS alongside the primary one running on the host system" [7]. This means that the average developer can run a separate operating system than the one installed on their machine, opening up a whole new world of potential efficiencies in development.

For example, many type 2 hypervisors feature powerful additional tools that can be installed into guest operating systems. One of these tools has gained immense popularity in recent years for enabling developers ways to design and manage their own development environments; a tool called Vagrant. Vagrant automates the configuration of virtual environments with other tools like Chef or Puppet, as well as enables the user to synchronize files between their host machine and virtual machines. Most developers will tell you that virtual machines are very convenient for development, but

not many want to actually edit files using a plain terminal-based editor over SSH.

Tools like Vagrant are the butter on the toast of type 2 hypervisors, but there is a reason that you don't see these hypervisors within server infrastructure. The hypervisor running as a process means that every action a VM takes must run through the host OS first, creating a potentially massive performance hit that essentially makes running several VMs incredibly inefficient. This is fine however, because the focus of this category of hypervisor isn't on performance, but rather the powerful ability of enabling the use of many operating systems at once.

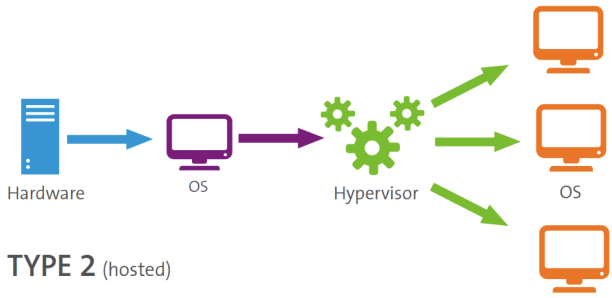

Figure 4: A Hosted Setup
[2]

There are many commercially available type 2 hypervisors today, the most popular (and free) of which are Virtualbox (by Oracle) and VMware Workstation. Both of these packages have a focus on user configuration of their VM, and since each virtual machine basically operates as it's own computer (hardware emulation, etc) there is a lot of interesting configuration to be done (see figure: 5). With these hosted VMMs, you can configure the amount of memory the machine has, the storage, shared folders, and most importantly the virtual Operating System. These VMMs are powerful tools to the wise developer, but also have been used in areas where it might be important to access applications only available on other software platforms, such as business users who need tools available on Windows while on a Mac (or vice versa).

One of the most important but least known practical use of type 2 hypervisors is that of computer security professionals. Virtual machines make it very easy to analyze malware without worrying about the corruption of your own machine, and if a failure were to occur you can simply rollback to a previous snapshot of the VM. Of course at this point most malware developers understand this simple fact, and a great deal of research has gone into developing malware analysis labs that can continue to study malware that itself is designed not to run within said labs [12].

Type 2 hypervisors have a much more practical use to the individual developer, and as such focus greatly on improving development efficiency, rather than computing efficiency as in bare-metal VMMs. While you would almost never see type 2 hypervisors within production, their use in ensuring reproducible development environments between developers, as well as the safety and stability offered by running code within a VM, more than justifies their existence. On

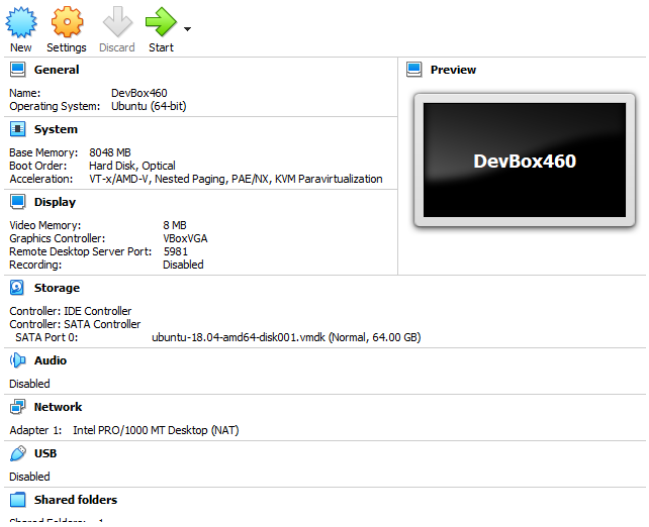

Figure 5: Virtual Box VM Manager

top of that, the ability to "demo" out new operating systems safely and without worry is incredibly useful.

With an understanding of the two different categories of hypervisors we can now dive into the details of their implementation, specifically in regards to how they interact with their host.

3.3 How Hypervisors Interact with their Host

The fundamental differences between Type 1 and 2 hypervisors are highlighted when looking at how they interact with their respective hosts. From a modeling prospective, a computer system can be abstractly visualized as having multiple layers. Each layer of the system 'houses' different components, some physical and some abstract. For example, consider a simplified personal computer: at layer 0 (L0) the hardware (Motherboard, CPU, Memory, etc.) is present, at layer 1 (L1) the operating system lives, and at layer 2 (L2) applications run. These layers not only represent a basic hierarchy of subsystems/components, but also represent an access-hierarchy, with each layer having access to the layer directly below it from a security/permission standpoint, see fig. 6 for a circular visualization of the same concept. In the simplified personal computer example, L2 (applications) would have access to L1 (the OS), but would not have direct access to L0 (hardware). Using this basic example of a personal computer, an examination of Type 1 and 2 hypervisors and where they fit in this hierarchy can be shown.

As previously mention, Type 1 Hypervisors are known as 'Bare Metal' hypervisors. This is because they are installed directly at L1 in place of the OS. As such, they reside directly on top of the physical hardware that's present at L0. From this position, the Type 1 Hypervisor can instantiate multiple Virtual machines (VMs), each with their own OS at L2. Furthermore, each OS can run applications at the next level up in the hierarchy (L3 and beyond). Because Type 1 Hypervisors have direct access to the underlying hardware without any competition for resources (such as OSes or device drivers) they are regarded as the most efficient and best performing hypervisors available [6].

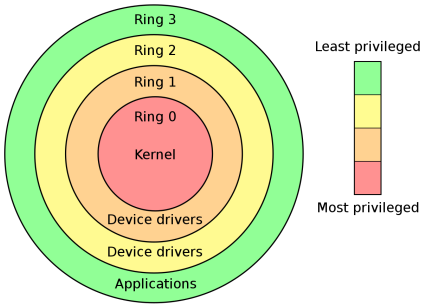

Figure 6: Security/Protection Visualization
[4]

The direct access that Type 1 Hypervisors have to the physical hardware dictates their interactions with the host machine. Since they interact directly with the system's CPU, memory, and physical storage, they are considered particularly secure. This is because there is nothing between them and the CPU for an attacker to compromise and the attack point of the OS has been removed. Type 1 Hypervisors operate in kernal space of a system because they take the role of a privileged OS Kernal.

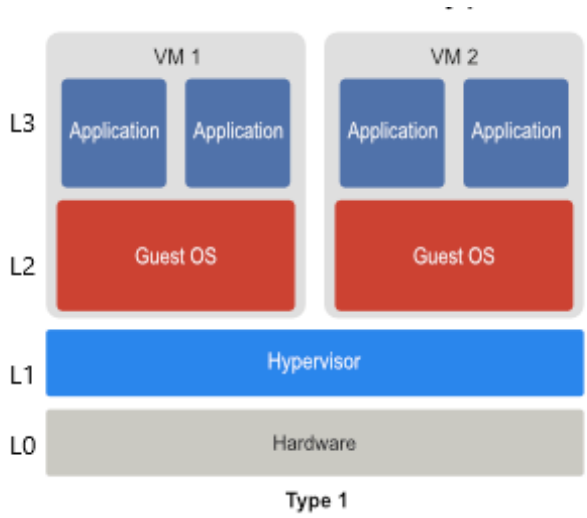

Figure 7: Type 1 Hypervisor Layer Visualization
[1]

Type 2 Hypervisors differ in their interactions with the host primarily because of their location in the system layer hierarchy. While Type 1 Hypervisors reside directly above L0, Type 2 Hypervisors reside above the OS at the L2 application layer. From this position, the Type 2 Hypervisor has similar abilities to the Type 1 Hypervisor, in that it can instantiate multiple VMs with their own OSes at the next level (L3), and have applications running above that (L4). One consideration to note is that because the Type 2 Hypervisor resides at the L2 application layer, the system can run additional applications alongside the hypervisor [6].

Because Type 2 Hypervisors reside at the L2 application layer, they rely on the host OS for access to computing, memory, and network resources. Because of this, there can be some latency issues, affecting performance. Also, because of the L1 operating system layer, there are higher security concerns with Type 2 Hypervisors because a compromised host OS gives access to all VMs on the system.

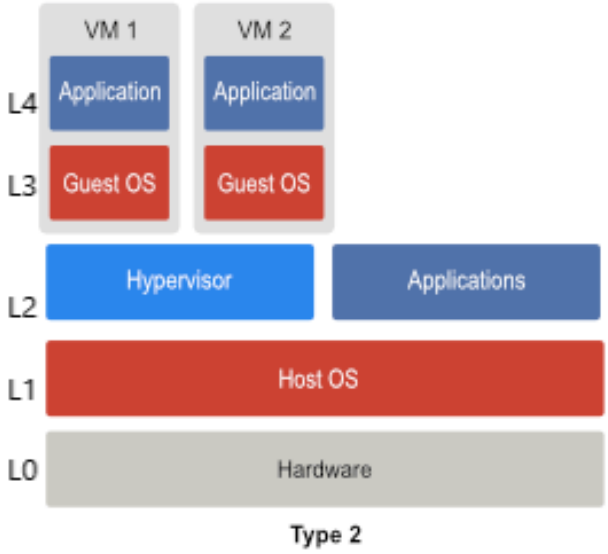

Figure 8: Type 2 Hypervisor Layer Visualization
[1]

3.4 Features of Hypervisors

Type 1 and 2 Hypervisors both act as a pseudo-operating system for virtualized systems. In this role, there are certain features that all hypervisors support.

Partitioning. A partition is a logical unit of isolation in which an operating system executes. Type 1 and 2 Hypervisors both partition the underlying hardware. Partitions generally do not have direct access to the physical processor and do not handle interrupts directly, rather they rely on the Hypervisor to handle interrupts and redirect them to the appropriate partition [5].

Resource Distribution. A major role of hypervisors is to fill the role of OS for virtualized systems. As such, hypervisors must schedule and distribute resources like memory, network bandwidth, etc. to the various virtual machines. Type 1 hypervisors fill this role identical to standard OSes, while Type 2 hypervisors must request resources from the host OS and then distribute those resources to the individual VMs.

Isolation. Simply put, isolation is the separation of one virtual machine from another. Hypervisors ensure this isolation to protect one virtual machine from a problem with another. This feature adds a layer of security between VMs because the hypervisor forms a virtual barrier.

Communication. Hypervisors also handle any communication between VMs. VMs will connect to one another through a virtual

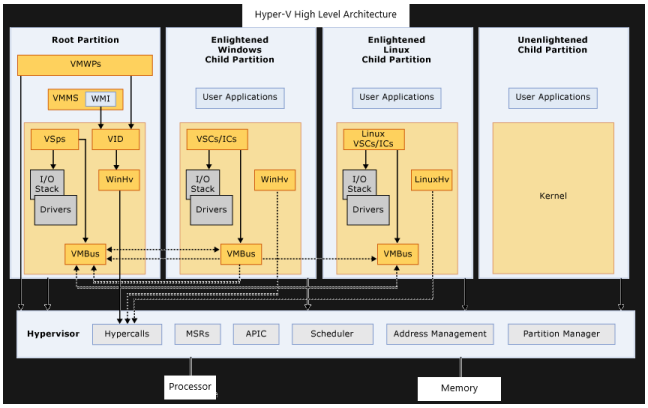

Figure 9: Microsoft's Hyper-V Partition Architecture [5]

network, allowing for file transfers. The hypervisor acts as a middleman, handling the transmission of messages from machine to machine.

While this list of features is not comprehensive, it does help clarify the role that hypervisors fill in a larger computer system. With these features, hypervisors excel at data replication (cloning virtual machines), consolidating servers, and desktop virtualization (particularly useful in development settings).

3.5 Analysis of Type 1 and Type 2

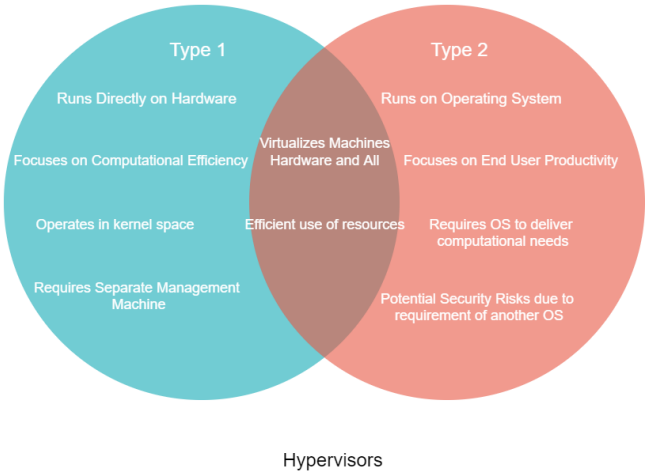

Figure 10: Comparison of Type 1 vs. Type 2

A few of the differences between type 1 and type 2 hypervisors were brought up in the last few sections, with a primary focus on how their implementation differs. In this section we hope to better compare and contrast the two different categories, as well as analyze their importance within the software development world.

As mentioned in the above sections, the primary difference between the two types of VMMs is that type 1 rests directly on top of the hardware while type 2 runs as a program within a user's Operating System. Type 1 hypervisors take the role of the operating system, running their controlled virtual machines much like another OS might run processes. In contrast, type 2 hypervisors are processes, running within the control of another operating system. This means that they are beholden to the limits of their host's OS, but most likely are more flexible and user friendly (see figure: 10).

This primary difference is very simple, but drastically changes how this software is utilized within the software community. Type 1 hypervisors were created for use in server infrastructure, and to this day they are used within IBM mainframes, network data centers, and countless servers across the globe. In contrast, Type 2 hypervisors were created much later as a system for bringing the power of virtualization to the end-user. These virtual machine managers are used by countless developers every day to power their virtual development environments, and continue to provide a whole new level of developmental efficiency.

Ultimately both of these categories really are just pieces of software that allows for abstraction from hardware. Whether it is used for server operating systems, virtual desktops, development environments, or studying malware; a hypervisor is a tool for efficiency.

3.6 Modern Advancements in Hypervisor Technology

Hypervisors have been around since the 1960's and are still used today but there are other virtualization techniques that might affect the use of hypervisors. Starting with the most popular, lightweight-weight virtualization is gaining more popularity as cloud computing grows. Containers are an outstanding technology for providing light-weight virtualization. They are however unable to provide the full isolation capabilities of virtual machines. Containers share the same host OS and have access to the OS system call interface this can lead to various security problems [8].

There is also server-less computing which is a new computing model described as Function as a Service (FaaS), which allows application developers to deploy functions instead of actual application. Operating systems have also started to adapt to house hypervisors better. Unikernels package the OS and the application into one bunch, which runs in the same CPU protection level [10].

4 DISCUSSION - A STUDENT PERSPECTIVE

The use of hypervisors has become more widespread as advancements in technology have pushed the capabilities of individual machines to new heights. For students studying Computer Science and related fields, this is interesting and valuable because hypervisors have become mainstream in the workforce.

Type 2 hypervisors, in particular, are more versatile and available to developers and students than ever before. Often, there is a need to develop a product or complete an assignment on a different OS than the developer/student's host system. System resources of modern computers are very abundant compared to even 10 years ago, so using hypervisors to create virtual machines with the needed OSes/applications has become a practical solution to this problem.

There are many popular hypervisor services available on the market for commercial and private use. This widespread availability implies that the technology is maturing and becoming a standard

service. As such, education on the history, application, and advancements of Type 1 and 2. Hypervisors can offer value to students and developers alike.

5 CONCLUSION AND FUTURE WORK

After conduction an in-depth research of hypervisors. We found that the hypervisor came around the 1960's under the umbrella of z/VM IBM's ground-breaking virtualization software for the system Z computer. There are two types 1 and type 2. Type 1 hypervisor is more efficient because it runs on the hardware of the hosted machine. Therefore, it is used in enterprise level development. Type 2 focuses more on the end-user and instead operate as an application within the operating system. They are easier to set up and are used more by individuals like students than enterprises. Most importantly, we learned how each type of hypervisor interacts with the host machine and its resources. Since type 1 hypervisors run directly on top of the hardware, they have direct access to resources hence their efficiency. They are also more secure than type 2 because they run directly on top of the hardware. Type 2 hypervisors rely on the host OS for resources. Both types of hypervisors do provide efficient use of resources and the ability to run multiple virtual machines on one system.

In the future we would like to conduct the same analysis with containers. We would also like to go as far as looking at the differences between containers and virtual machines. Containers have gained a fair amount of popularity and we would like to learn more about them and what makes them so popular.

REFERENCES

[1] [IMAGE]Type 1 and 2 Hypervisor Layers. 2019. . Retrieved November 10, 2020 from https://www.vembu.com/blog/type-1-and-type-2-hypervisor

[2] [IMAGE]What Does a Hypervisor Do? 2014. . Retrieved November 10, 2020 from https://www.flexiant.com/2014/02/05/what-does-a-hypervisor-do/

[3] Bill Bitner. 2012. *VM – A Brief Review of Its 40 Year History.* Retrieved November 9, 2020 from http://www.vm.ibm.com/vm40hist.pdf

[4] [IMAGE]Wikipedia Commons. 2010. . Retrieved November 10, 2020 from https://commons.wikimedia.org/wiki/File:Priv_rings.svg

[5] Microsoft Documentation. 2018. *Hyper-V Architecture.* Retrieved November 13, 2020 from https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/reference/hyper-v-architecture

[6] Michael Eder. 2015. *Hypervisor-vs. Container-Based Virtualization.* Retrieved November 10, 2020 from https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2016-07-1/NET-2016-07-1_01.pdf

[7] IMB Cloud Education. 2019. *Hypervisors.* Retrieved November 11, 2020 from www.ibm.com/cloud/learn/hypervisors.https://www.ibm.com/cloud/learn/hypervisors#:~:text=A

[8] Manco et al. 2017. *My VM is Lighter (and Safer) than your Container.* Retrieved November 13, 2020 from https://doi.org/10.1145/3132747.3132763

[9] Michael; et al. Finn. [n.d.]. *An Evaluation of KVM for Use in Cloud Computing.* Retrieved November 10, 2020 from https://ci.coastal.edu/~mmurphy2/files/papers/icvci08.pdf

[10] Madhavapeddy. 2013. *Unikernels: library operating systems for the cloud.* Retrieved November 13, 2020 from http://dx.doi.org/10.1145/2451116.2451167

[11] Shannon Meier. 2008. *IBM Systems Virtualization: Server, Storage, and Software.* Retrieved November 12, 2020 from http://www.redbooks.ibm.com/redpapers/pdfs/redp4396.pdf

[12] Adrian Sanabria. 2007. *Malware Analysis: Environment Design and Archtitecture.* Retrieved November 11, 2020 from https://www.sans.org/reading-room/whitepapers/threats/malware-analysis-environment-design-artitecture-1841