Fall 2021 CS 687 Capstone Project Final Report Containerization on an Example Closed System Architecture

Adam Coffey
Advisor: Dr. Sion Yoon
MS in Computer Science
School of Technology & Computing (STC)
City University of Seattle (CityU)

Abstract

The current system is overly complicated due to legacy architecture. Containerization can be used to solve this issue by converting many of the virtual machines into containers, reducing the number of operating systems and the usage of system resources. Containerization could be accomplished using Docker to containerize the applications so that they perform the same function without a virtual machine. This would make the system easier to install, update, and maintain. This project compared the installation and usage of a host's resources by a simple application on a virtual machine and the same application on a container to provide a proof of containerization's advantages.

Keywords: Containerization, Virtualization, System Architecture, Virtual Machine, Docker

1. INTRODUCTION

This project is an attempt to use containerization to improve the virtualization of an example closed system architecture. This closed system architecture consists of a hypervisor on a server that hosts several virtual machines. These virtual machines include the DNS server, the logging server, the workstations, and several application servers.

Problem Statement

The current architecture has several virtual machines that only serve one purpose. This is because of a legacy system that splits up the functions of the system into different machines. Those physical machines have now become virtual machines, each with its own application or applications. This means there is an operating system installed onto a virtual machine for each application. Additionally, each of those operating systems is required to have other overhead to meet requirement standards, such as security software. Finally, because this is a closed system installed at several locations, the additional steps required to install and configure the operating systems for each virtual machine complicate the installation procedure.

A potential solution to this issue is containerization. With containerization, the architecture could be simplified by eliminating several of the virtual machines and replacing them with containers that run only the application and its dependencies. This would eliminate several operating systems as well as the software required on them. This could also potentially mean an improvement to the installation process as containers can be made into small ISO files that would fit on an installation media, removing the operating system installation and configuration steps.

Motivation

There are potentially two problems that affect the stakeholders: the users, the maintainers, and the installers.

The first is the problem that the system is overly complicated. Making the system simpler would mean that it is easier to maintain because a simpler system has fewer points of failure and fewer parts to check when debugging. For maintainers, this means an easier job, and for users, this means more system uptime.

The second problem is that the installation is overly complicated. Making virtual machines into containers would lower the number of steps required to get the application up and running. This would make the job easier for an installer attempting to install the system onto new hardware or a maintainer that is reinstalling the system to fix an issue.

Approach

Containerization is typically done by installing the application onto a host operating system and then running one of the containerization tools, such as Docker, LNX, or Simplicity, to turn the application into a container. Afterward, that container is installed into the system to run the application without the host operating system.

The approach for this project will be the same. Virtual machines will be installed onto a hypervisor, and then applications will be installed onto those hosts. There will be basic metrics captured to view the resources used by the virtual machine and the application. The containerization will be performed, and the containers will replace the virtual machines. Finally, more basic metrics will be captured to determine if the containers are running the application with less resource usage.

Conclusions

The expected outcome is that the containers perform the same function as the virtual machines. The containers will use less of hypervisor resources and be easier to get running after a shutdown. The containers will have smaller installation files, be simpler to install and be easier to configure.

2. BACKGROUND

Virtualization is the method of turning a computer system running on hardware into a virtual machine that runs on a hypervisor. This virtual machine appears identical to the computer system on hardware but has only a portion of the hardware’s memory, storage, and networking. The hypervisor itself is software that is put onto hardware such as a host server or computer that manages the resources of that host to divide them up and give them to the virtual machines it hosts.

Containerization is a newer technology that expands on virtualization. It is the method of packaging just the application and its dependencies into a container that can be placed onto a host. However, they are not as robust and user-friendly as virtual machines, so they require a good application to work correctly.

3. RELATED WORK

The topic that is being researched is Containerization in an Example Closed System, so it has focused on containerization, with emphasis on its application. The sources found have included several explanations of the concepts of virtualization and containerization, which are related and very important for the project. It has also introduced several tools that could be used for containerization, including LXC and Docker. It has also included troubleshooting and improving containers. Finally, it included the use of containers in system architectures to help with the understanding of how to include it in this project.

Literature Review

The issue that is attempting to be fixed is that the current system is overly complicated and has unnecessary overhead, which could be fixed with containers. It is a closed system that includes several virtual machines that have only one application running on them. This could be the perfect opportunity to convert some of those virtual machines to containers. A containerization tool would need to be selected, and the major applications would need to have their virtual machines converted to containers. Afterward, it would need to demonstrate that it performs the same or better than the previous architecture.

Because this is a conversion from virtualization to containerization, a comparison done by M. J. Scheepers is a particularly valuable starting point. This paper compares the two topics by introducing both a virtualization tool and containerization tool, giving descriptions of how both are used, and then comparing how they perform actions such as SQL queries (Scheepers, 2014).

There are a couple of extra benefits to containerization that are uncovered in the research. One study by Zhang, Vasilescu, Wang, and Filkov discussed a benefit of containerization in that it can help with continuous deployment by allowing for seamless updates (Zhang et al., 2018). This is an advantage that helps with system uptime. Another paper discussed how containers are used in mobile environments with an emphasis on how it provides security benefits by protecting the application and preventing the application from growing outside of the container (Oluwatami, Midi, & Bertino, 2017). Though the mobile architecture is only tangentially related to the topic, the added security benefits will help to validate it.

A few papers serve as introductions to containerization tools. One introduced Singularity as it attempted to improve the implementation of the application AutoDock Vina by containerizing it to skip compilation, giving the benefits of making the app faster and lowering the number of dependencies (Hisle, Meier, & Toth, 2018). It not only introduced the tool but also included the Linux commands used in their implementation. Another introduced NetContainer as a tool that can make more efficient containers for some architectures (Hu, Song, & Li, 2017). It gave a tutorial on NetContainer and how to implement it in a particular architecture. Yet another paper introduced Linux and FreeBSD containers by comparing them and providing metrics on the benefits of each (Puig, Villalobos, Rodero, & Parashar, 2017). It had a few tools that might be useful as well as their application in closed system architecture.

There were several sources in the research that included containerization in different architectures. One paper by Stytz, Adams, Garcia, Sheasby, and Zurita included the usage of containers in several environments and how they fit into their overall systems (Stytz et al., 1997). They applied containers to solve several different problems in several different systems.

A paper by C. Pahl discussed containerization as it is used in cloud applications. This paper is helpful for understanding how containers can fit into an architecture while also introducing the LXC containerization tool (Pahl, 2015).

Another paper discussed how the authors attempted to add a position-based application to a mobile container architecture and also discussed how to interface with and modify containers (Oluwatimi & Bertino, 2018). This paper is helpful because it included their attempts to customize containers.

One paper discussed how they use containerization in supercomputers and how they had to make their IO interface more efficient (Huang, Wang, Lu, & Liu, 2021). The introduction of container IO issues is an interesting disadvantage that will need to inform the architecture.

Another paper that attempts to improve an architecture with computers in a specialized environment is one on edge computing by Zhang, Zhou, Ge, Wang, & Hwang. This paper includes an application of containers and experiments on those containers to make them more efficient in their edge computing environment, with metrics to back up their experiments (Zhang et al., 2021).

Yet another paper that introduces containerization in a different environment is one on containerization in fog computing. This source discusses how they utilized Docker to create containers that communicate with each other, all on a raspberry pi (Bellavista & Zanni, 2017). Their application of containers in a closed system with an emphasis on communication between containers is very similar to this paper.

Review Conclusions

Containers are a new and widespread technology that provides an alternative to virtual machines. Containerization can be used in some system architectures to improve some metrics and reduce the overhead of the system. There are also additional benefits, including improved security and an increased amount of system uptime. However, there may be IO problems or a reduction in efficiency of some operations if the containerization is not done carefully.

There are several tools in widespread use, including Docker, LXC, Singularity, and NetContainer. They can be used in Windows, Linux, FreeBSD, and mixed systems to containerize an application.

Containerization is used in many architectures in many different types of systems. This includes cloud applications, supercomputers, edge computing, and fog computing. Containerization brings benefits to each of those types of systems but requires different approaches and has different hurdles to overcome.

Containerization should be helpful in the closed system that is being improved. It should allow for the conversion of virtual machines to containers and provide similar metrics with less resource usage.

4. APPROACH

User Requirements

The container solution needs to perform the same functions at the same level as the virtual machine solution. It also needs to show less resource usage of the host to justify the transition cost. It must also be as simple or simpler to install than the virtual machines.

Design

The design of the container solution will be very similar to the existing virtual machine solution, but with some virtual machines replaced with containers. Figure 1 below is a visualization of the existing system, which is an eSXI Hypervisor Host managed by VMware VCenter. On this host are individual virtual machines that serve as a DNS server, several user workstations, a SQL server, and hosts for applications.

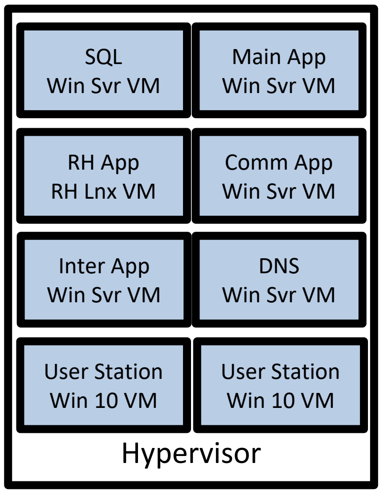

Figure 1: Virtual Machine Architecture

The SQL Server and application virtual machines exist only to host the SQL and applications, so they will be converted into containers, as visualized below in Figure 2.

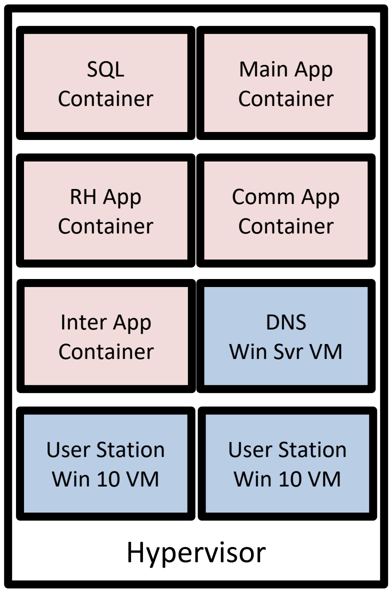

Figure 2: Container Architecture

Implementation

To achieve a partial implementation, one virtual machine with a sample application and one Docker image of the same sample application was used. This allows for a partial proof of concept that could be expanded to the full system with more time and resources.

VMware Player was used as the host for the virtual machine implementation on a desktop computer. A virtual machine was created on that program using the Windows Server 2019 ISO image. On that basic image, Tomcat, Java Runtime Environment, and Java Software Development Kit were installed to support the sample application. Finally, the sample application was placed in Tomcat's file structure, and the application was run.

Docker was used for the container implementation, also on a desktop computer. The Docker container image was created by creating a Dockerfile with the appropriate code calling on Tomcat and running the Docker build command. Finally, the Docker run command was run on the Docker container image to achieve the same application result as the Virtual Machine implementation.

Both applications worked by assigning them a web address on the localhost. They were simple web pages that allowed the user to navigate to a couple of other web pages.

Technologies Used

Docker was used as the containerization software. Docker is a software for Windows, Linux, and Mac that allows for the creation of a container around an application or even a series of linked containers containing several applications.

VMware Player was used as the host for the virtual machine. VMware Player is the free version of VMware Workstation and allows for virtual machines to be hosted on desktop computers, managing the memory, CPU, disk, and network usage and separation from the host.

Windows Server 2019 is an operating system created by Microsoft. It includes many features not available in the Windows desktop version that are managed using the Server Manager, including more support for SQL Server.

The sample application used is a simple Tomcat application made by Tomcat with a Java Server Page and a Java servlet page to test those capabilities (Apache, n.d.). This application was chosen because it is a Java based web application and simple to implement.

5. DATA COLLECTION

Data collection was performed by gathering baseline data on the virtual machine and on the container. Windows Performance Monitor was used on the virtual machine, while Docker's Performance page was used for the container. All the metrics change from moment to moment as different processes are active, or the application performs different actions, so the data was captured at their highest observed values. This was the highest observed set of values that was achieved by opening the application and causing a page to load.

The results are in Table 1:

	Virtual Machine	Container
CPU	26%	5.04%
Memory	1.3 GB	178 MB
Network I/O	113 KB/s	5.1 KB/s
Disk	11 GB	124 KB

Table 1: Virtual Machine and Container Metrics

6. DATA ANALYSIS

The four measured metrics are CPU, Memory, Network I/O, and Size. Each of these is important because they are hardware resources being taken up by the application and its host. Lowering any or all of them means a possible increase in performance for the host.

When moving from a virtual machine to a container, CPU was reduced by to a fifth, while Memory, Network I/O, and Disk were all reduced by an order of magnitude. All these changes can be attributed to removing the operating system and its applications.

7. FINDING

The findings are organized into two sections. The first section is the findings on the performance of the container versus the virtual machine, based on the data collected as discussed above. The second section is findings on the ease of installation for the container versus the virtual machine.

Performance
There is a reduction in usage in each of the four metrics captured when going from a virtual machine to a container. This is a good result to demonstrate that virtual machines that are used only for one application should be converted into containers. Additionally, the Virtual Machine numbers would be higher if additional required applications such as security software were added.

Each of the four components measured could be the bottleneck that benefits from reduction. Memory, Network I/O, and Disk are all an order of magnitude smaller in the container, meaning that the whole system could fit on a smaller server, or a duplicate set of containers could be placed on the server for redundancy and still be smaller than the virtual machine option.

Ease of Installation
This was not covered in the data collection or analysis, but the installation is made quicker, simpler, and easier when switching from a virtual machine to a container for this sample Tomcat Java application.

The virtual machine instance of the application required the installation of the operating system, the installation of Java, and the installation of Tomcat, followed finally by the installation and configuration of the application. None of these steps were particularly difficult but would become more complex with more complex applications.

The container instance of the application required that a Dockerfile with three lines of configuration be placed in the folder with the application and then that the Docker create and run commands be run from the command line. This requires knowledge of the Docker application, but after the process is made by a knowledgeable person, it would be easy to pass on to future installers.

The container installation required much less time because it did not need to install an operating system, which took up almost all the installation time for the virtual machine. It also has fewer steps to perform, as noted above. These steps are also easier because the virtual machine version requires more configuration of both the operating system as well as the application.

8. CONCLUSION

Containerization is a great improvement over virtualization in closed system architecture when applied correctly. Virtual machines that are used only as hosts for applications can be replaced with containers to reduce the load on the host and make the installation and maintenance easier. The conversion in the case performed in this project significantly reduced CPU, Memory, Network I/O, and Disk usages. It also reduced the number of installation steps by skipping operating system and dependency installations. Finally, it makes maintenance easier by having less parts that can fail.

9. FUTURE WORK

This project could be expanded and improved upon by performing a larger and more thorough implementation. This implementation would include not only one conversion of a virtual machine to a container but the conversion of several in a closed system. This would match the blueprint written in the design section and require a series of applications that interact with each other and a SQL server.

This could be done by installing VMware ESXi on a server that has enough resources to host several virtual machines. On that VMware ESXi hypervisor, the Windows Server, RedHat Linux, and Windows 10 virtual machines could be installed. Applications could be installed on those Windows Server and Linux virtual machines. After ensuring those applications are working and communicating correctly, metrics could be taken on the performance of the full system. Finally, Docker could be installed on one of the Windows Servers, and the virtual machines hosting applications could be converted to containers. After once again ensuring the applications are working and communicating correctly, metrics could be taken for comparison.

This implementation would be a much more thorough proof that containerization would work for the example closed system by checking that there are no issues when having many containers with applications, such as the I/O issue mentioned in the related works section.

The project could also be improved and expanded upon by exploring other containerization options and containerizing more complex applications. Other containerization options may fit the closed system architecture better than Docker, and it would be useful to explore all of the options. Additionally, the sample application used is a simple Java Tomcat application that doesn’t communicate outside of the virtual machine. A larger, more complex application that communicates with a database would improve the project and be a better proof.

10. REFERENCE

Bellavista, P., & Zanni, A. (2017, January). Feasibility of fog computing deployment based on docker containerization over raspberrypi. In Proceedings of the 18th international conference on distributed computing and networking (pp. 1-10).

Hisle, M. S., Meier, M. S., & Toth, D. M. (2018). Accelerating autodock vina with containerization. In *Proceedings of the Practice and Experience on Advanced Research Computing* (pp. 1-5).

Hu, Y., Song, M., & Li, T. (2017, April). Towards" Full Containerization" in Containerized Network Function Virtualization. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems* (pp. 467-481).

Huang, L., Wang, Y., Lu, C. Y., & Liu, S. (2021). Best practice of IO workload management in containerized environments on supercomputers. In Practice and Experience in Advanced Research Computing (pp. 1-7).

Oluwatimi, O., & Bertino, E. (2018, March). A multi-enterprise containerization approach with an interoperable position-based system. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy* (pp. 256-266).

Oluwatimi, O., Midi, D., & Bertino, E. (2017). Overview of mobile containerization approaches and open research directions. *IEEE Security & Privacy*, *15*(1), 22-31.

Pahl, C. (2015). Containerization and the paas cloud. *IEEE Cloud Computing*, *2*(3), 24-31.

Puig, F. X., Villalobos, J. J., Rodero, I., & Parashar, M. (2017, December). Exploring the Potential of FreeBSD Virtualization in Containerized Environments. In Proceedings of the10th International Conference on Utility and Cloud Computing (pp. 191-192).

Scheepers, M. J. (2014, June). Virtualization and containerization of application infrastructure: A comparison. In 21st twente student conference on IT (Vol. 21).

Stytz, M. R., Adams, T., Garcia, B., Sheasby, S. M., & Zurita, B. (1997). Rapid prototyping for distributed virtual environments. *IEEE Software*, *14*(5), 83-92.

Apache (n.d.). Sample Application. Retrieved from: https://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/

Zhang, J., Zhou, X., Ge, T., Wang, X., & Hwang, T. (2021). Joint Task Scheduling and Containerizing for Efficient Edge Computing. IEEE Transactions on Parallel and Distributed Systems, 32(8), 2086-2100.

Zhang, Y., Vasilescu, B., Wang, H., & Filkov, V. (2018, October). One size does not fit all: an empirical study of containerized continuous deployment workflows. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 295-306).

10. Appendix

A demonstration of how the project's virtual machine and container implementations were performed was uploaded to YouTube. The video is titled Containerization on an Example Closed System Demo, and the link is below:

https://www.youtube.com/watch?v=uvESwZ0lFdQ