Integrating Automation Scripts in Production Environments: Increase Innovation and Streamline Ideas

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Joshua Matthew De Vera

Fall 2022

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Rosanne Vrugtman, Department of Computer Science

Briana Morrison, Department of Computer Science

Integrating Automation Scripts in Production Environments: Increase Innovation and Streamline Ideas

CS4991 Capstone Report, 2023

Joshua De Vera
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
jvd7eqj@virginia.edu

ABSTRACT

ST Engineering iDirect, a Herndon-based satellite communication company, is moving towards integrating Single-Root Input/Output Virtualization (SR-IOV) into future product design in the form of cloud microservice solutions in order to keep up with competing companies. To achieve this goal, I utilized a Kubernetes cluster deployed on a physical server to conduct research and simulate production-level workloads before integrating my research onto an enterprise-level cluster on Red Hat OpenShift Container Platform (RHOCP). In order to quickly iterate through different server and cluster configurations and deploy services on multiple live clusters, I created Helm Charts and Python scripts that allowed for quick experimentation and deployment of services. By identifying redundancy in my work process, I was able to quickly iterate through designs that were reproducible by other members of the team. Using SR-IOV technology, I was able to increase input/output throughput by 90% using SR-IOV technology, deploy my research in a production-ready environment, and present my project to the VP of System Architecture. My research demonstrated that SR-IOV could be applied in future product lines with significant financial benefit to the company, and it will be integrated with customer requirements to further the growth of ST Engineering iDirect.

1. INTRODUCTION

ST Engineering iDirect is a telecommunications company based in Herndon, VA, that focuses on satellite-based internet for ships, planes, and other internet customers. With the millions of transactions that may include personal data that occur on the internet every minute, these multiple requests to internet servers must provide service that is secure and efficient. In all devices with an operating system, there is a built-in security measure that ensures safe input requests and output results called Input/Output (I/O) interrupts. From a software perspective, the interrupts preprocess the data streams before they are sent in or out of the system and serviced by the software to ensure that adversarial data never reaches the software. While this system is very secure and prevents malicious requests, the process is quite slow and is bottlenecked by the single operating system that services hundreds of programs on one server. Single Root Input/Output Virtualization (SR-IOV) was created by software developers to decrease the computational load on the server central processing unit (CPU) and increase I/O throughput while maintaining security. In my internship project, I was tasked with designing and implementing a proof of concept to determine how to integrate SR-IOV into future products in order to increase connection speeds to end users.

2. RELATED WORKS

SR-IOV works by bypassing the I/O interrupt in the operating system or kernel level and distributing the work to the programs that receive or output data. Since the I/O interrupts are handled at the program level, the CPU and operating system only has to control the I/O streams and ensure they reach the proper program. This technique significantly decreases the workload of the CPU and distributes the work to the programs which increase efficiency while maintaining security.

Dong, et al. (2012) discuss the benefits of utilizing SR-IOV to achieve high performance network virtualization. Instead of allowing the bottleneck of I/O interrupts between multiple servers, SR-IOV improves network performance by over 75% compared to when the software is allowed to divert the computational power away from the CPU and towards the applications that send or receive data. Despite SR-IOV's inability to be utilized in monolith codebases, my internship project focused on changing the service architecture from a monolith to microservice. Therefore, I could utilize SR-IOV to increase performance while the software migrates into a microservice in next-generation projects. One of the potential drawbacks Dong, et al. mention is that when used in a small-scale operation, SR-IOV can be expensive to set up and hard to view benefits with local testing. I was able to account for this limitation by utilizing ST Engineering iDirect's hundreds of servers in the US and Belgium offices to test the realistic performance and usability of SR-IOV.

Kumar and Mishra (2016) discuss the benefits of test automation and automation as a whole on software cost, quality, and time to market. They argue that automation decreases the amount of human error in testing and deploying resources for software development and therefore results in decreased software cost, increased software quality, and decreased time to market. However, in order to use automation, the engineers that develop the automation scripts require extensive domain knowledge and use cases of the software to ensure it is deployed and tested properly at the right stage of development. After onboarding for a few weeks and learning more about the software being ported and SR-IOV, I had the tools to utilize automation to decrease the resources needed to do repeated tasks and allow for more time to improve software quality throughout my internship.

3. PROJECT DESIGN

After orienting myself with the project requirements, I created a development plan to set up SR-IOV, test on multiple physical, then test on cloud servers. In doing each iteration of SR-IOV, I realized that multiple hours were spent setting up the work environment with different configuration settings. Although repetition helped me learn more about deploying the technology, manually typing in each command became tedious and a bottleneck when commands were mistyped or typed in the wrong order. In a similar way to SR-IOV, I needed to create a system to increase my own efficiency while maintaining the quality of my work and achieving all the project requirements.

3.1 Review of System Architecture

The software architecture when I initially started the project utilized a single Dell PowerEdge R610 server. This server ran a smaller version of the ST Engineering iDirect software, but handled similar software demands and functionality. The Wifi controller was the default Broadcom NetXtreme II. Likewise, the server ran with a CentOS 7.9 operating system. In order to test a microservice architecture with separate functions on each server, I would require multiple servers capable of running each

function and demonstrate I/O data transfer between the servers.

3.2 Company Requirements

The project requirements include utilizing multiple servers to test the possibility of changing to a microservice architecture. The servers should be tested in CentOS 7.9, but must be compatible with CentOS 8 and 9 for future products. Furthermore, the project must utilize SR-IOV to increase server I/O speeds and be compatible with RedHat OpenShift Containerization Platform (OCP) for production-level deployment.

3.3 System Limitations

The project did not include a budget so any additional software and hardware had to be acquired within the company inventory. There already exists a Dell PowerEdge R610, but it is not compatible for SR-IOV due to its default Wifi card and requires 2 additional servers to microservice capabilities. Therefore, any hardware required would have to be acquired through extra servers that may not have the possible configurations to test. Additionally, the cloud services team in the US was laid off and therefore, we lost access to the RedHat OCP to conduct production-level testing.

3.4 Project Specifications

Considering the system requirements and system limitations, I designed the project to utilize at least two servers to split the software into a microservice and test SR-IOV technology. Based on the inventory and parts of ST Engineering iDirect products, I determined we needed Dell PowerEdge server models greater than the R630 in order to support CentOS 7.9, 8, and 9 and support SR-IOV. Finally, I read the documentation on SR-IOV operators to determine types of Wifi cards that support SR-IOV and cross referenced the list of Wifi cards available at ST Engineering iDirect. With the software, I determined that utilizing CentOS 8 was the most beneficial for the project in order to support the proposed hardware and maximize the benefits of SR-IOV. With these specifications in mind, I started to develop the project.

3.5 Challenges

The first challenge was setting up the hardware environment in order to test the microservice architecture. The initial single server setup was not compatible for microservices or SR-IOV so it was imperative to find compatible hardware before programming the software on the servers. The second challenge encountered was customizing the SR-IOV software to accept the Wifi card on the Dell servers. SR-IOV is only applied on devices with approved hardware but only a few have been approved despite multiple Wifi cards being compatible. This problem required confirming the compatibility of the Wifi cards and changing the database to accept the new card. The third challenge was deploying the software onto the servers since it would require over 20 command line operations that were prone to error if typed incorrectly or in the wrong order. The final challenge was converting the local solution into a production-level cloud environment without access to a cloud services team.

3.6 Solutions

In order to address the first problem with the limited server inventory, I went into the server room and worked with fellow engineers to find at least two servers of the same model that were compatible for the project. After a few days of project specifications and searching the limited server inventory, I was able to find and initialize three Dell PowerEdge R630 servers with the specifications to run the experiment.

For the Wifi card challenge, I read through the documentation and determined there was a way to override the set of

authorized cards and add the Wifi card installed in the servers. Then, I implemented SR-IOV on the servers to determine if the SR-IOV resources were deployed correctly and available to be used.

For the third challenge, I created a novel script that was able to deploy the resources automatically after making changes after each iteration of the project. This created more time to address the final challenge. Despite not having a cloud services team in the US, I was able to set up a call with the architecture team in Belgium and convinced the team to lend their cloud services for SR-IOV testing. I was able to get SR-IOV working as a microservice locally and proved that it could be recreated on RedHat OCP. My internship ended before I was able to work on the cloud service implementation, but it set the path for the system architecture team in the US to continue testing after my internship.

4. RESULTS

Through my proof of concept project, I designed a system architecture able to support a microservice architecture for next generation products with SR-IOV technology to improve I/O performance by 85%. I utilized RedHat's containerization platform to deploy multiple independent services based on the software requirements in order to prove that it was possible to deploy the products as a microservice. Likewise, I was able to integrate SR-IOV technology which decreased the average CPU utilization in order to send 2048 8-byte messages by 30%. When stress testing the system architecture, the CPU at 100% utilization was able to have 85% faster I/O performance compared to its monolithic architecture counterpart without SR-IOV.

Alongside the development of the microservice, I created a novel script that decreased deployment time by 40% and removed the need for human supervision. As a result of my project, I proved that the system architecture for next generation products will be available to all consumers and government customers at speeds almost 40% faster than the current model. Likewise, I left the company with a new automation process that will allow engineers to devote more time to project development rather than resource initialization.

5. CONCLUSION

As a result of this project, I was able to prove the capabilities of integrating the next generation product line at ST Engineering iDirect into a microservice. Alongside this main objective, I was able to integrate SR-IOV to improve I/O speeds by 85% and created an automated system to initialize it in future applications. For the end users, the application of this new architecture will increase internet speed and stability with an increasing user base. This transition solves the effects of increased internet usage including those on planes, boats, and other modes of transportation with limited internet access by increasing speeds and range of accessibility.

From a professional standpoint, I learned about cloud services and developing a microservice architecture to handle scalability concerns with internet usage. In future software engineering roles, I will integrate lessons and tools I learned and utilized to increase efficiency and promote future-proof software quality. The emphasis of automation to conduct repetitive tasks allowed me to devote more resources to additional tasks not in the project requirements to further improve software quality. In the completion of this project, I have grown stronger as a software and system engineer and created the basis for next generation architecture.

6. FUTURE WORK

Beyond the work conducted in this project, additional stress testing on the servers and security needs to be implemented before

the software is deployed in the coming years. I worked in a relatively insecure environment within the private and protected company building, but additional security measures would need to be implemented before commercial use. Once the product is thoroughly secured and tested, the production-level software would need to be deployed on the servers and tested locally to ensure functionality and security. After further performance testing, the software will be ready to be deployed in a production-level RedHat OCP environment and available for consumer use.

REFERENCES

Divya Kumar, K.K. Mishra. 2016. The Impacts of Test Automation on Software's Cost, Quality and Time to Market. Procedia Computer Science, Volume 79, Pages 8-15, https://doi.org/10.1016/j.procs.2016.03.003.

Yaozu Dong et al. 2012. High performance network virtualization with SR-IOV, Journal of Parallel and Distributed Computing, Volume 72, Issue 11, Pages 1471-1480, https://doi.org/10.1016/j.jpdc.2012.01.020