Monitoring Resources and Snapshots on VMware Infrastructure

Prathiba Lakshmi Narayan[1], Siddharth Shankar Das[2], S Gopi[3], Kritika Agrawal[4], R Suganya[5]

{prathiba642@gmail.com[1], siddharthshankardas3@gmail.com[2], sgopis.suresh@gmail.com[3]}

School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, India [1,2,3,4,5]

Abstract. In today's resource-intensive virtualized environments, efficient management of resources is crucial to ensureoptimal performance and prevent resource bottlenecks. When the resources are scaled, managing and monitoring them becomes difficult therefore, there comes a need of automating managing, monitoring, and alerting and taking precautionary measures in order to save the hardware from degrading or damaging. There are instances when the hardware is under-utilized or over-utilized, in both scenarios the hardware is not being optimally used. This paper presents a comprehensive approach to monitor VMware resources using the Go programming language and GOVCtool. We utilized VMware ESXi servers as the virtualization platform and vCenter to create a centralized datacenter. Wepresent the architecture, implementation details including all attempts to emulate a datacenter, and results of our system,to successfully automate snapshot deletion. Our solution serves as a valuable tool for administrators and organizations seeking to streamline resource management in VMware ESXi-based datacenters.

Keywords: VMware, ESXi, Snapshots, Virtualization, Data Center, Go.

1 Introduction

We were given the opportunity to work on this problem statement through Hewlett Packard Enterprise's Catch Them Young 2023 program and Vellore Institute of Technology. The 5 of us were amongst the 25 students selected for this program from our college.

Virtualization has revolutionized the world of datacenter management, enabling organizations to achieve significant efficiency gains, resource consolidation, and agility in deploying new services. VMware's ESXi server virtualization platform, combined with vCenter for centralized management, is a well known datacenter infrastructure. However, with the growing complexity of virtualized environments and the ever-increasing demand for computing resources, effective resource monitoring, and management have become challenges [1].

IACIDS 2023, November 23-25, Lavasa, India

DOI 10.4108/eai.23-11-2023.2343224

This paper presents a novel solution that uses the Go programming language to monitor and manage resources in VMware ESXi-based datacenters. Our approach provides a snapshot management system to mitigate storage overhead.

We begin by introducing the core components of our solution, detailing how we use the capabilities of ESXi servers and the centralized control provided by vCenter to create a scalable and efficient datacenter environment, while also talking about the challenges faced along the way in implementing the datacenter. We then developed a tool to address the challenge of snapshot management, which is vital for VM backup and recovery but can lead to resource waste if not managed properly. Our automated snapshot management system periodically assesses snapshot usage for each VM. Upon exceeding a predefined threshold, the system notifies the user of the same and initiates the automatic deletion of older snapshots, optimizing storage resources and reducing the risk of resource contention.

This paper discusses the existing solutions in this field, our proposed architecture, how we implemented our solution with screenshots showing the working and how it improves the performance. We then discussed the various challenges we faced while implementing it and how we overcame the same.

2 Literature Review

In the context of our internship, our team was entrusted with the responsibility of developing a solution to address critical challenges faced by Hewlett Packard Enterprise (HPE) and other companies that rely on virtual machines (VMs) as their primary computing infrastructure. After thorough investigation, we found out that these challenges were rooted in the unchecked growth and overuse of snapshots in VM-based systems, which had far-reaching implications for resource management, performance, data integrity, and cost-efficiency. The paper "VMCSnap: Taking Snapshots of Virtual Machine Cluster with Memory Deduplication" by Yumei Huang et al.[2] presents a novel snapshot strategy for virtual clusters. It effectively reduces snapshot size and I/O bottlenecks by discovering and storing duplicate memory pages just once.In our research, we referenced "Comparison between common virtualization solutions: VMware Workstation, Hyper-V, and Docker" by Ziyu Li [3]. This study found that working on VMware offers superior hardware efficiency in different operating systems compared to other virtualization platforms.

Wenting Wei et al. [4] focus on addressing resource management challenges in cloud computing, particularly in network virtualization. They emphasize the importance of network virtualization for enhancing flexibility and scalability in cloud services. The paper discusses efficient allocation of multi-dimensional physical resources in data center servers.Hoi Chan et al. [5] address ensuring high availability in virtualized environments with a "smart adaptive snapshot replication technique." This technique leverages existing VM infrastructure and common utilities to provide high availability within virtualized environments. JongBeom Lim et al. [6] present a distributed snapshot protocol for efficient AI computation in cloud computing. The paper tackles the challenge of capturing the global state of resources to reduce processing time in case of failures and enable AI applications to resume from the latest snapshot. Bo Li et al. [7] address challenges in distributed snapshots within Virtual Machine Clusters

(VMCs) in IaaS cloud environments. Their innovative approach, eHotSnap, decouples coordination from snapshots and optimizes memory snapshots to enhance efficiency.

2.1 Comparing existing methods with proposed method

In their comparative analysis of snapshot management techniques [8], Abhinav et al. introduced Trusted VM Snapshots (TVM), a hypervisor-based system aimed at ensuring snapshot consistency in untrusted cloud infrastructures. TVM accomplishes this by intercepting guest Direct Memory Access (DMA) operations and using a Copy-on-Write (CoW) approach to protect memory pages during snapshot creation. It addresses security concerns and provides a snapshot generation and verification protocol. However, our proposed snapshot management method for VMware ESXi environments offers significant advancements.

Our method enhances snapshot management within the ESXi virtualized environment through intelligent automation and resource optimization. Unlike TVM, our approach focuses on minimizing disruptions and expediting snapshot creation processes. By optimizing resource allocation and utilization, we ensure that snapshot operations don't hinder the overall performance of the host system or other running VMs.

What sets our solution apart is its simplicity of implementation. In contrast to TVM, which necessitates hypervisor-level modifications and integration with additional security mechanisms, our method takes a more straightforward and user-friendly approach. Its user interface simplifies configuration and monitoring, making snapshot management more accessible for administrators. Moreover, our solution seamlessly integrates with the broader VMware ecosystem, ensuring compatibility with various VMware products and services.

While in the paper “Towards an efficient snapshot approach for virtual machines in clouds” Jianxin Li et al. [9] have efficiently saved the snapshots by incorporating the feature of skip page saving, but none of them till now have given an efficient approach for managing the deletion of snapshots.

As such, our primary objective became clear: to devise a robust solution that would effectively manage snapshots and mitigate the issues arising from their excessive proliferation. This paper presents our comprehensive analysis, design, and implementation of a snapshot management solution, addressing the pressing concerns of HPE and other industry leaders in the realm of VM-based systems.

3 Architecture

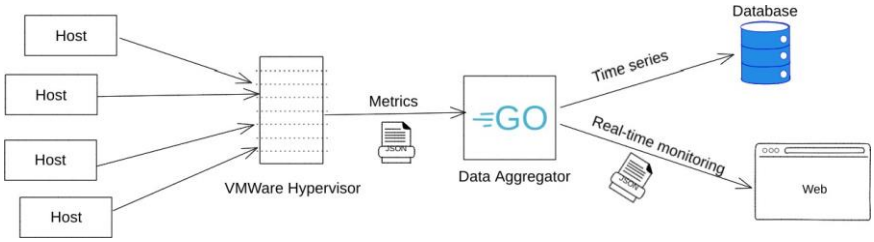

Fig. 1. Initial Architecture

4 Implementation

Since this project is a hardware intensive project, we needed systems that had good hardware specifications. The university systems that we used had the following configuration -

RAM - 64 GB; HDD - 2 TB HDD; CPU - i7 11th Gen Intel Processor; GPU - 12 GB NVidia; OS - Linux Mint Cinnamon

During the entire course of the project, we referred to the Udemy course 'The Complete VMware vSphere 7: Beginner to Advanced-part ½' [10] and 'The Complete VMware vSphere 7: Beginner to Advanced-part 2/2' [11] by Fettah Ben.

4.1 Phase I

Our first task was to achieve virtualization with ESXi 6.7 which is a type 1 hypervisor. This was set up on VMWare Workstation 16 Pro. We configured the IPV4 as well. The ESXi can now be accessed through this address to add virtual machines. In the ESXi environment, the installation of a minimum of five virtual machines (VMs) was carried out with the primary goal of closely monitoring and analyzing their resource usage.

The utilization of a thin provisioning process was employed during the VM installations, enabling efficient storage allocation and utilization. To enhance resource visibility and facilitate accurate measurements, each individual VM was allocated a minimum of 10 GB of storage. We noticed that the CPU usage, memory and storage data of each virtual machine was already visible to the user. With these valuable insights, and a few other graphs, we were able to find another use case of our project which was to delve into virtual machine snapshots.

Initially, our ESXi setup encountered a challenge, as the allocated RAM proved insufficient for the intended number of concurrently running virtual machines on each ESXi host. Our original choice was to install Windows 7 and Windows 8 as the operating systems for the VMs, but it became evident that lighter OS alternatives were necessary for optimal performance. Consequently, we made the switch to LinuxMint, a more resource-efficient option. We found that crucial information about the virtual machines, along with informative graphs, was readily available, thereby already monitoring VM resources. We then decided to move into a new use case, exploring snapshot management. Using the guidance from our mentor, we initiated automated snapshot deletion procedures, thereby enhancing the efficiency and management of our virtual infrastructure.

4.2 Phase 2

We then proceed to emulate a datacenter with VMWare VCenter 6.7. Our successful attempt involved using Windows Servers to function as vCenter Server and Domain Controller. We installed 2 VMs running Windows Server 2012 on VMWare Workstation 16 Pro. We named one Window Server, which would function as the DC, as winDC and configured its network properties, i.e., set a static IPv4 address of 172.16.224.11. Then, we added Active Directory Domain Service and Domain Name System roles and winDC was added to a new AD Forest and created a new user for the Windows Server running vCenter Server.

Now, on the other Windows Server, on which vCenter Server would be installed, we modified its network properties by setting its static IPv4 address as 172.16.224.12. Then, we named it as

wincenter and added it to the newly created AD Domain using the user credentials created in winDC and restarted wincenter.

vCenter Server 6.7 ISO image was mounted and then installed and deployed using Embedded Deployment on the Windows Server named wincenter with default port settings and custom Single Sign-On configuration.

The virtual switch present in VMWare Workstation 16 Pro would facilitate all communication between the vCenter Server (wincenter), the DC (winDc), the ESXi hosts and the base Linux machine. Now, we were able to successfully access the vCenter Server through its vSphere Client on wincenter, winDC and the base Linux machine too.

We then created a new Datacenter using the vSphere Client of the vCenter Server and added our 2 ESXi hosts to the datacenter. We now have complete access to monitor and manage each ESXi host and their underlying VMs to perform various actions regarding Power, Snapshots, etc.

4.3 Phase 3

Finally, we needed to implement our use case which was to manage the virtual machines and their snapshots using Go and GoVC [12]. We installed Go and GoVC on our systems then set up environment variables with the VCenter URL, the username and password. We then tested some GoVC commands like ‘govc about’ to ensure that we can access the Vcenter.

We then tried some datacenter specific commands from the govmomi - govc documentation [13] like “govc datacenter.info" that gives us the number of hosts, clusters, virtual machines and datastores. “govc vm.info <vm_name>” gives us details about a particular virtual machine such as its name, memory, cpus, its host URL and whether it is powered on or off. When dealing with snapshots more specifically, we use the command “govc snapshot.tree -vm <vm_name> [options]”. Different options give us the snapshot name, snapshot ID, date of creation and snapshot size. This is the information we require to implement our use case. We extracted the useful data from this and stored it in a data structure.

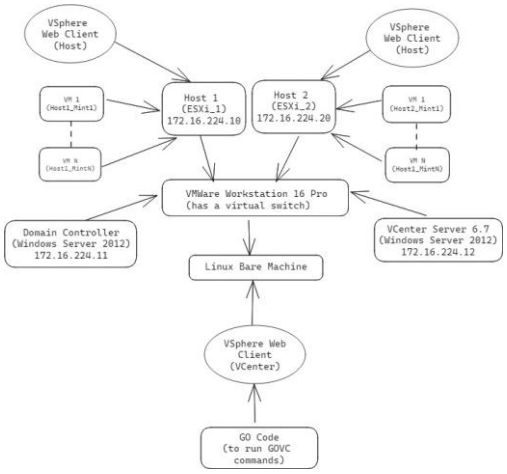

Fig. 2. Final Architecture with all three phases

4.3.1 Explaination of code functions

A structure named Snapshot is defined to store individual snapshot details. It has four fields: name (string), id (string), size (float64), and date (time.Time).

The sizeToMB function takes a size parameter of type string, representing the size of a snapshot. It converts the size from either kilobytes (KB) or gigabytes (GB) to megabytes (MB) and returns the result as a float64 value. This is done to unify the size in order to judge all the snapshots and derive a conclusion on how much time shall the snapshot be stored in the server.

The currentDate function returns the current date as a time.Time value, formatted in the same way as the creation dates of the snapshots. time.Now() is called to obtain the current date and time. This function returns the current local time. date.In(loc) is called to change the timezone of date to UTC.

The snapLife function calculates the age of a snapshot given its creationDate compared to the current date. snap_lifespan is a variable that stores the difference between the current date (obtained by calling the currentDate function) and the creationDate. The Sub method returns a time.Duration representing the time interval between the two dates.

The next function getVMSnapdetails is to get the details of the snapshot from the specific virtual machines. Each command represents a variation of the "govc snapshot.tree" command with different arguments (-i, -s, -D) to retrieve snapshot details.

The storeSnapDetails function populates a slice of Snapshot structs with snapshot information. It processes lines containing details like ID, name, size, or creation date. If the detail is "name," it trims whitespace and assigns it to the snapshot's name field. For other details, it extracts values enclosed in square brackets. If the detail is "size," it converts thevalue to megabytes using the sizeToMB function. If it's "crDate," it parses the value as a time according to a predefined layout. This function effectively organizes and stores snapshot details, facilitating easy access and management.

Functions for displaying and deleting snapshot details from the structure have been created. The checkSnapshots function identifies "illegal" snapshots based on criteria, considering size and age. It iterates through snapshots, calculating days since creation with snapLife. For different size thresholds, it checks snapshot age: over 5120 MB, age > 3 days; between 1024 MB and 5120 MB, age > 30 days; under 1024 MB, age > 180 days. "Illegal" snapshots are tracked in the illegalSnaps array. This function efficiently identifies snapshots needing attention due to size and age.

The takeAction function acts on "illegal" snapshots. If the action is "delete," it uses "govc snapshot.remove" to delete the snapshot, capturing and printing the removal output. The snapshot is removed from the structure. If not "delete," it warns that the snapshot will be deleted after 5 days. This function automates snapshot management by either immediate removalor issuing a warning about impending deletion.

5 Results

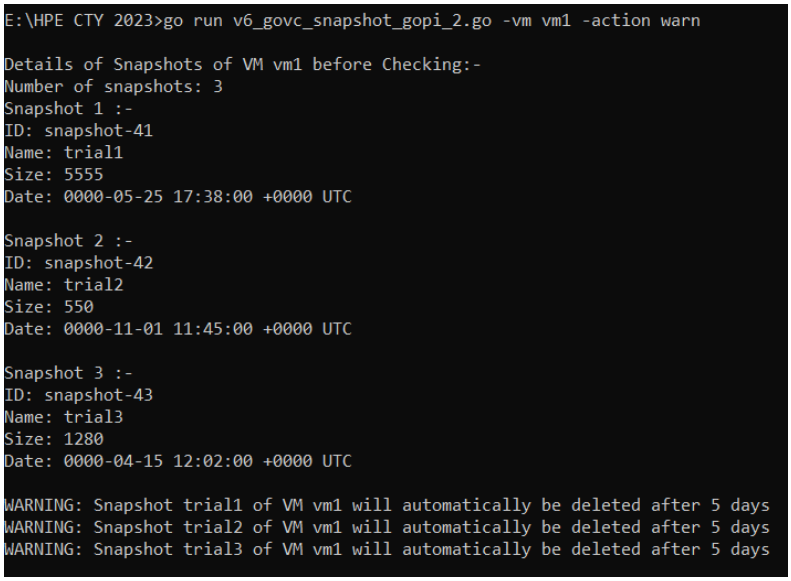

Fig. 3. Working of code showing warnings to the user

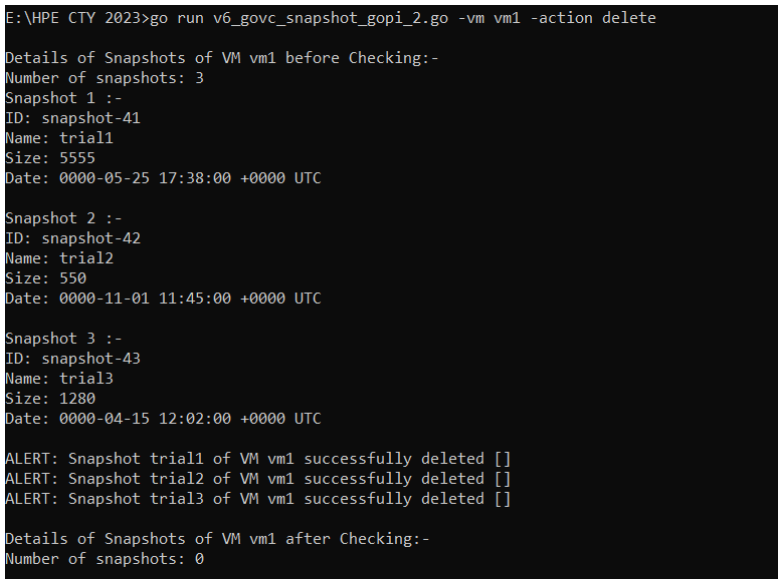

Fig. 4. Working of code showing deletion of snapshots

This implies that every virtual machine only consumes resources that it requires since snapshots that are not required are automatically deleted. This further implies that these resources can be used by other systems that require the resources. Cost for companies using virtualized environments that pay as per their usage is also reduced when lesser resources are being used.

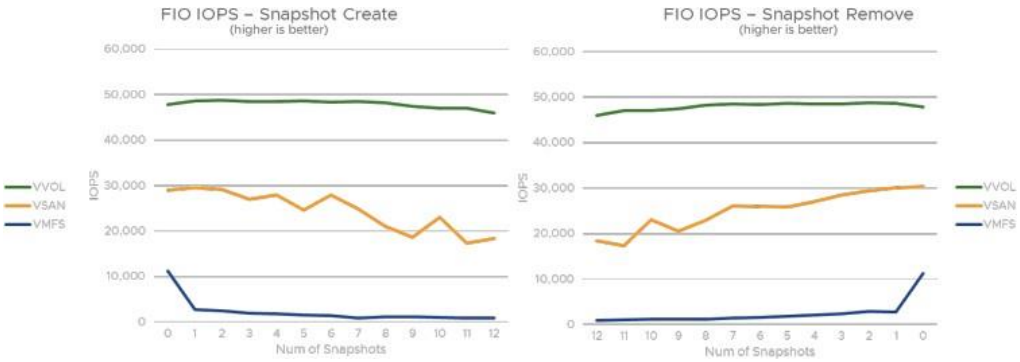

Fig. 5. IOPS performance with snapshot create and delete [14]

As seen from the graphs above from VMware's documentation on "VMware vSphere Snapshots: Performance and Best Practices" [14] lesser number of snapshots also implies better performance, faster application response times (Input/Output Operations Per Second) and improved overall efficiency.

6 Discussion

In phase 2, before successfully emulating the datacenter, we had a few unsuccessful attempts that lead us to the successful implementation, which we will brief about below.

Our first attempt involved installing the vCenter Server Appliance on a Linux environment. We faced a couple of challenges during the stage 2 of the deployment process. To overcome this challenge, we meticulously analyzed the error log files, aiming to identify and resolve the underlying causes. We even tried different configurations/settings (like Tiny, Small, experimenting with network configuration like ip, fqdn, dns, etc) while installation, but they were all unsuccessful.

In our second attempt, we tried installing VCSA on our Laptops and connected them using a Wi-Fi router to ensure that network limitations of our university systems would not hinder our progress. We successfully completed stage 2 of the installation. We could now access the VSphere Client. But, we could not proceed with this setup as our laptops did not have enough RAM to run multiple virtual machines.

In our next attempt, we tried to install VCSA in a Windows environment in our university systems. In this Windows environment (lab), the ESXi hosts and even VCSA were easily installed without any hurdles and we successfully managed to emulate a datacenter. This led us

to believe that there are some kinds of network restrictions in our previous lab (Linux environment) which were interfering with the deployment of VCSA.

7 Conclusion and Futrue Work

This project has shown us how to effectively use Go and GoVC to monitor virtual machine resources through ESXi hosts within a Vcenter managed datacenter. The inclusion of real-time alerts and automatic deletion of snapshots based on predefined thresholds enhanced system administrators' ability to proactively manage resource allocation and prevent performance bottlenecks. It also shows the different challenges one might face while setting up a data center. As more and more organizations rely on virtualization for their services, our project lays the groundwork for further advancements in resource optimization and management within virtualization.

As future work, we plan to do the final stage of our proposed architecture which was to store the collected data and present it in the form of a dashboard where users and organizations can manage their snapshots through the dashboard interface.

Acknowledgments. We extend our heartfelt gratitude to our esteemed institution Vellore Institute of Technology, for providing us with the invaluable opportunity to undertake this project. The access to the state-of-the-art lab systems and resources greatly contributed to the successful completion of our endeavor. This would not have been possible without VIT Chennai's support.

We would also like to express our sincere appreciation to Hewlett Packard Enterprise (HPE) for offering us an incredible opportunity to collaborate and immerse ourselves in the corporate world. The exposure to cutting-edge technologies and hands-on experience in the real-world applications of virtualization significantly enriched our learning journey.

We extend our thanks to our dedicated mentor from HPE, Mr. Akash Deo, whose guidance, insights, and expertise were pivotal in shaping our project. Their unwavering support and encouragement propelled us forward, helping us bridge the gap between theory and practical application. This project would not have been possible without the collective support, encouragement, and collaboration of these entities. We are truly grateful for their contributions.

References

[1] A. Iqbal, C. Pattinson and A. -L. Kor, "Performance monitoring of Virtual Machines (VMs) of type I and II hypervisors with SNMPv3," 2015 World Congress on Sustainable Technologies (WCST), London, UK, 2015, pp. 98-99, doi: 10.1109/WCST.2015.7415127.

[2] Huang, Yumei & Yang, Renyu & Cui, Lei & Wo, Tianyu & Hu, Chunming. (2014). "VMCSnap: Taking Snapshots of Virtual Machine Cluster with Memory Deduplication." Proceedings - IEEE 8th International Symposium on Service Oriented System Engineering, SOSE 2014. 314-319. 10.1109/SOSE.2014.45.

[3] Z. Li, "Comparison between common virtualization solutions: VMware Workstation, Hyper-V and Docker," 2021 IEEE 3rd International Conference on Frontiers Technology of Information and

Computer (ICFTIC), Greenville, SC, USA, 2021, pp. 701-707, doi: 10.1109/ICFTIC54370.2021.9647226.
[4] Wenting Wei, Kun Wang, Kexin Wang, Huaxi Gu, Hong Shen, Multi-resource balance optimization for virtual machine placement in cloud data centers, Computers & Electrical Engineering, Volume 88, 2020, 106866, ISSN 0045-7906, https://doi.org/10.1016/j.compeleceng.2020.106866.
[5] Hoi Chan and Trieu Chieu. 2012. An approach to high availability for cloud servers with snapshot mechanism. In Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference (MIDDLEWARE '12). Association for Computing Machinery, New York, NY, USA, Article 6, 1–6. https://doi.org/10.1145/2405146.2405152.
[6] Lim, J.; Gil, J.-M.; Yu, H. A Distributed Snapshot Protocol for Efficient Artificial Intelligence Computation in Cloud Computing Environments. Symmetry 2018, 10, 30. https://doi.org/10.3390/sym10010030.
[7] Bo Li, Lei Cui, Zhiyu Hao, Lun Li, Yungji Liu, Yongnan Li, "eHotSnap: An Efficient and Hot Distributed Snapshots System for Virtual Machine Cluster" in IEEE Transactions on Parallel & Distributed Systems, vol. 34, no. 08, pp. 2433-2447, 2023. doi: 10.1109/TPDS.2023.3272014
[8] Srivastava, A., Raj, H., Giffin, J., England, P. (2012). Trusted VM Snapshots in Untrusted Cloud Infrastructures. In: Balzarotti, D., Stolfo, S.J., Cova, M. (eds) Research in Attacks, Intrusions, and Defenses. RAID 2012. Lecture Notes in Computer Science, vol 7462. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-33338-5_1.
[9] Jianxin Li, Yangyang Zhang, Jingsheng Zheng, Hanqing Liu, Bo Li, Jinpeng Huai, “Towards an efficient snapshot approach for virtual machines in clouds”, Information Sciences, Volume 379, 2017, Pages 3-22, ISSN 0020-0255, https://doi.org/10.1016/j.ins.2016.08.008.
[11] Fettah Ben, "VMware vSphere: Install, Configure, Manage," Udemy, https://www.udemy.com/course/vmware-vsphere-2/.
[12] Fabian Lee, "VMware: Using the govc CLI to Automate vCenter Commands," Fabian Lee's Blog, https://fabianlee.org/2019/03/09/vmware-using-the-govc-cli-to-automate-vcenter-commands/
[13] VMware, "govc - Usage," GitHub Repository: vmware/govmomi, https://github.com/vmware/govmomi/blob/main/govc/USAG.
[14] VMware, “Performance Best Practices for VMware Snapshots”, https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/performance/vsphere-vm-snapshots-perf.pdf.