International Journal of
Intelligent Systems and Applications in Engineering

ISSN:2147-6799 www.ijisae.org Review Paper

Impact of Virtualization Technologies in the Development and Management of Cloud Applications

A.S.Gowri*[1], Dr.P.Shanthi Bala[2]

Submitted: 16.11.2018 Accepted : 19.04.2019

Abstract: Today most of the consumer services ranging from education to banking, hospital management to ticket booking are made online. The online services are hosted in cloud and they are mostly time-critical applications. The cloud-based applications depend on datacenter (DC) resources for computation, communication, and storage. The resource utilization in the cloud needs to cope with the dynamic workload and stochastic request spikes. Virtualization is the key technology for effective resource utilization in the cloud data center. The type of virtualization technology (VT) that is adapted for the delivery of cloud application ensures the quality of service. The goal of this paper is to compare and contrast the performance measures of various virtualization technologies for heterogeneous workloads. The paper presents the impact of VT in the development of application in the cloud. Each virtualization technology outperforms the other in some or other performance metrics. In spite of the differences, certain virtualization technology dominates depending upon the application requirements in the software development sector.

Keywords*: Virtual machines, Docker Containers, Hypervisor, Kubernetes, Lightweight VMs, Microservices, Unikernel.*

1. Introduction

Cloud computing provides value added computing environment in terms of resources like servers, networks, storage, and software. Right from the start-up companies to the veteran software developer, depend on the cloud resources to develop and deploy applications on a large scale. The applications that are hosted in the cloud to cater its service on pay per usage are referred to as cloud services. Facebook,salesforce.com, YouTube, and Twitter are some of the services managed in the cloud [1].

The cloud guarantees the provision of resources for the applications that are hosted in its DC [3]. The huge demand for cloud resources drives the Cloud Service Providers (CSP) to choose the right technology [2]. Virtualization is the key solution that can yield efficient resource utilization with minimal cost and reduced energy consumption.

Google search engine supports search operation in different locations around the world. The search activity rises and falls in each data centre according to the time of the day and events affecting various part of the world. Miles Ward[1], Global Head of Google Cloud revealed that containerization is one of the secrets to the speed and smooth operation of the Google search engine. He expressed that Google Search operation, launches nearly 7,000 containers every second, which amounts to about two billion per week. The significance of container usage in such a large scale motivates the study of VT in comparison with its counterpart.

The remaining section of the paper is organized as follows. Section2 gives background details about the various virtualization technologies. Section 3 discusses the related works conducted in virtualization, Section 4 describes the performance of the VT in cloud application environment and Section 5 conclude with future perspectives.

[1] *Research Scholar, Pondicherry University, Puducherry, India. ORCID ID: 0000-0002-4193-4358*
[2] *Assistant Professor, Pondicherry University, Puducherry, India, ORCID ID: 0000-0003-0576-4424*
** Corresponding Author Email: sivakgowri@gmail.com*

2. Background Details

With the emergence of big data, there is a demand for the huge storage and compute facility [7]. Cloud DC is the ultimate solution for the compute/storage infrastructure at a large scale [1]. The consumers can avoid capital expenditure[2] (Capex) and operational expenditure (Opex) by leasing the compute/storage servers from clouds. Capex refers to the initial investment cost spent on setting up the compute facility like cost of hardware, software, company space, staffs etc. Opex refers to the cost to be spent on maintenance in the long run. The CSPs leverage the VT for server consolidation and efficient resource utilization to improve their revenue/server ratio [2] [8].

Virtualization is the process that emulates Virtual machines/ Virtual containers on a physical host machine. A number of Virtual machines/Virtual containers are invoked in a host machine to achieve server consolidation [1]. Server consolidation exploits the host to its fullest and leads to fewer machines usage. Usage of fewer machines (resources) in turn, reduces the cost and energy consumption of the data-centre. Hence choosing the right VT profits the CSPs as more computing power can be extracted at the cost of less number of machines.

The different kinds of VT that are implemented in DCs can be categorized as hardware-based virtualization and OS based virtualization [1][2]. Virtual machines(VM), Virtual Containers (VCs), Containers within VM, Lightweight VMs and Unikernels [2] are the various types of VT. These virtualization technologies come with their own deployment and orchestration frameworks. OpenStack, Vcenter [4] serves an example for virtual machine orchestration framework whereas Docker Swarm and Google's kubernetes are the orchestration frameworks for Virtual Containers [8].The following section discusses virtualized and containerized resources along with their variants.

[1] **https://www.informationweek.com**
[2] **https://www.2ndwatch.com/cloud-computing-shift-from-capex-to-opex**

2.1. Hypervisor-Based Virtualization

Hypervisor-based virtualization is also referred to as Hardware Virtualization or Bare Metal Virtualization. A virtual machine is a representation of a real machine that is emulated by a hypervisor[3]. More than one VM can be instantiated in a physical machine. Each VM owns its own guest operating system along with its respective drivers, binaries, libraries to build and run the application. Just like a physical host, more than one application can be executed in a VM. In short, each VM behaves as though it is a separate physical host with no awareness about the existence of other VMs.

A Hypervisor or a Virtual Machine Monitor (VMM), is a software (SW) that creates and run multiple VMs per host [4]. The VMM virtualizes the host server and sits in between the hardware and the Virtual Machine. The hypervisor can be distinguished as Type1 and Type2 hypervisor as shown in **Fig.1(a),(b)** respectively [2,5,11]**.** In Type1, the hypervisor is mounted on the bare metal (Hardware infrastructure – CPU, RAM, Disk, NIC) over which the VMs are emulated. KVM, VMware ESXi, Xen are the examples for Type1 Hypervisor.

In Type2, the hypervisor is mounted over the host OS of the physical machine. The VMs that are emulated over the hypervisor reserves its own share of physical resources. Oracles' virtual box is an example of Type 2 hypervisor. In either type, each VM emulated in a host run its own guest OS. One or more applications can be run on each VM with their respective library files and dependencies**.**

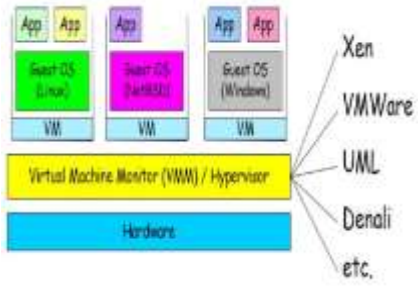

Fig.1.(a) Type1- Bare Metal Hypervisor

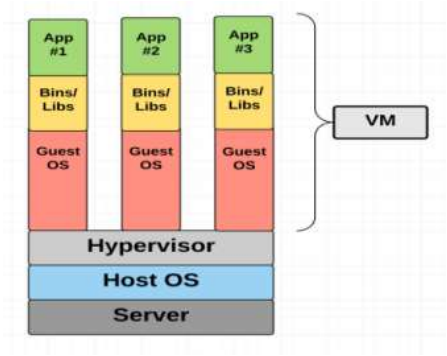

Fig.1.(b) Type 2-Host OS Hypervisor

2.2. OS Based Virtualization

The OS-based virtualization also called container virtualization is shown in F**ig.2 (a), (b)** [11]**.** It involves virtualizing the OS kernel rather than the physical hardware. The OS-based virtualization encapsulates the OS and their dependencies into containers which are collectively managed by the underlying OS kernel [4]. The virtual container (VC) permits multiple isolated user-space instances to run on a single physical host. System containers and Application containers are the two types of virtual containers [2] [11]. Linux Containers (LXC) is an example of a system container. They run all types of system processes like initializing, directory commands etc. Whereas the application containers run applications only [5]. Docker engine otherwise called a container engine is an example of application containers and its execution process is called Dockerization [11].

The Open source platform docker engine automates the process of deploying, shipping and running distributed applications within container [5] ie. The docker container packs up the application code, system runtime tools, libraries and drivers into a single image (code portability) that can be installed and executed on any server [6]. To be crisp, in container virtualization, the container engine decides how much resources (CPU, memory, disk, NW) are to be allocated to the containers [8]. Cloud services deployed using containers takes less execution time when compared to VMs as it takes time to boot the Guest OS along with its dependencies. But VMs provide the highest degree of isolation to its applications that lack with containers.

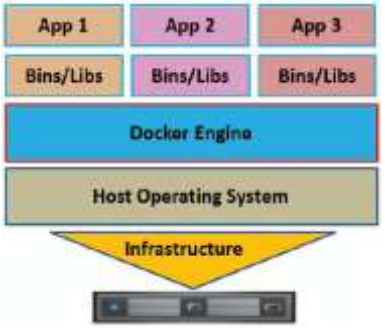

Fig.2. (a) Containers in Linux based machines

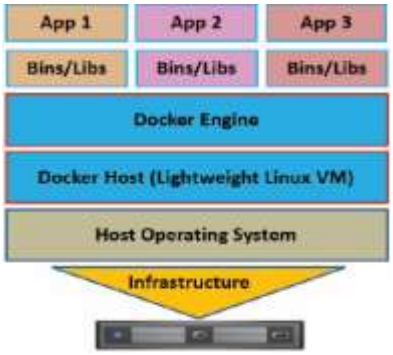

Fig.2.(b) Containers in Non-Linux based machines.

2.3. Containers Within VM

The container within VM technology is shown in **Fig.3 (a)** [5].As the VCs are inside VM, the technology serves a double purpose. First, minimal execution time is achieved through VC. Second, through the VMs, isolation, and security is guaranteed to the application deployed in the VC. Amazon Web Services (AWS) Elastic Container Services (AWS ECS) and Google Container Platform (GCP) container engine run containers within VM instances.

Nesting containers within VMs havetheir own significance too [4]. It costs less for the DCs to reuse the already existing VM technology. Also, the gaps in each of the technology can be overcome by the other thus leading to better performance. Hence, Amazon and Google have adopted the container within VM technology for good reasons. The technology is considered as the reason for Google's popularity. Container within VM is a boon for the public clouds that need to run an application concerning security with less cost.

2.4. Light Weight VMs

The key idea behind Lightweight VMs lies in "Speed of the Containers and Security of the VM" [4]. Instead of nesting containers within VM, the lightweight approach employs low overhead hardware virtualization [6]. The VM runs a customized

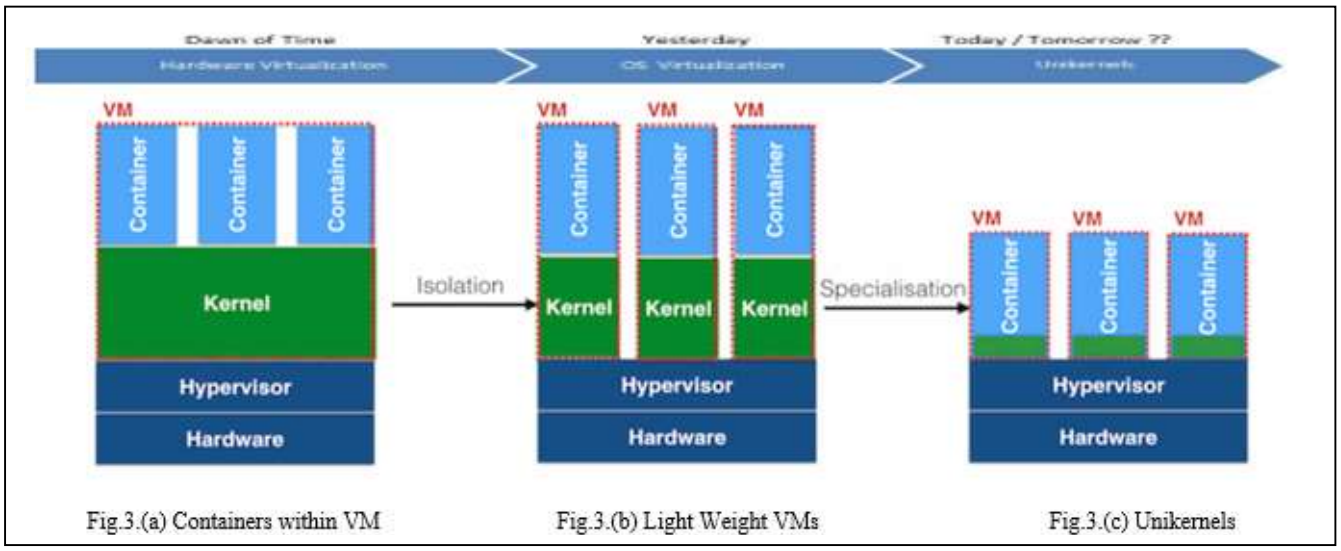

Fig.3.(a) Containers within VM Fig.3.(b) Light Weight VMs Fig.3.(c) Unikernels

kernel which possesses quick boot up time and low memory size consumption, hence called as lightweight VMs as in **Fig.3 (b)** [4]. Vector Linux, clear Linux, TinyCore and VMware's Bonneville are the examples of the Lightweight VMs.

Lightweight VM addresses two issues of conventional VMs viz., host transparency and Footprint [4]. Footprint refers to the memory consumption to instantiate the VM. The VM footprint is minimized by discarding redundant functionality of the hypervisor. The lightweight VM boots up in one second rather than ten seconds of traditional VM. While it takes 0.3 seconds for lightweight VM than 0.8 seconds of Containers [4]. Regarding host transparency, lightweight VMs directly access files (library, executable etc.,) on the host's file system without getting transferred to virtual disk as in the case of traditional VMs.

2.5. Unikernel

A unikernel is an executable image that can execute natively on a hypervisor, without the need for a separate operating system. The image contains application code and all the operating system functions required by that application. It is a single user, single process applications (no threads, fork or multiuser) embedding the full application stack [5]. When deployed on the top of the hypervisor, they benefit from being light, ultra compact and results in better performance [2]. Unikernels are sealed against modification once deployed in a cloud platform [18].

It is motivated by the fact that what it would be like, if all the software layers in the applications were compiled within the same framework. In Unikernels as in **Fig.4 (a), (b)** [18], the application images are built with only those OS components that they actually require. Unikernels are the single process applications that assure high performance, fast boot and small attack surface (secure). Unikernels best suit for applications like video streaming-delivered through cloud [19] where latency is of primary importance.

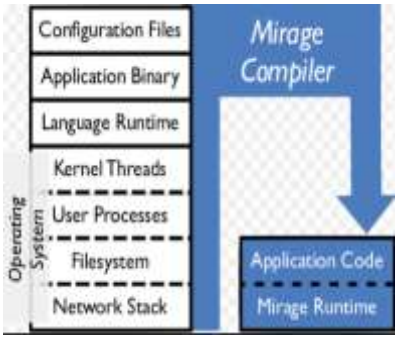

Fig.4. (a) Unikernel Configuration

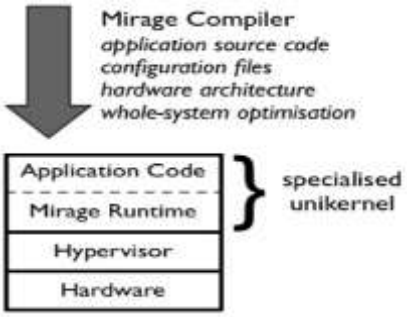

Fig.4. (b) Unikernel

3. Related Works

The rapid growth in DCs urges the need for server consolidation to optimize resource utilization [1]. Virtualization is the key to achieve server consolidation where the resources of the underlying physical host are multiplexed to emulate different VMs [4]. Hence more applications are packed per physical host reducing the number of physical machine usage in the cloud. This section discusses the various works done in virtualization technology.

Mardan et al. compared containers and VMs for handling disk I/O in DBMS. When compared with LXC, KVM outperforms without compromising isolation. The file system journaling in containers lead to poor performance in LXC [1]. The performance of bare metal, VM and VC are evaluated in OpenStack. Though bare metal outperforms VM which in turn proved better than VC, the goal to

Table1. Pros and Cons of Virtualization Technology

Technologies	Advantages	Disadvantages
Virtual Machines	Isolation, Security at system level [14]	Increased cost and time for instantiation, migration.
Virtual Containers	Lightweight, smaller footprint, less cost and time to instantiate. Isolation at the application level	Less secure, low networking bandwidth and scalability, performance interference.[4]
Containers inside VMs	Increased Isolation, security at both system and application level. Minimal Migration time and latency.	Increased time to instantiate and boot
Light Weight VMs	Highly kernel dependent, less boot time [20].	Single purpose applications alone served better [18].
Unikernels	High Isolation, security, smallest footprint, portability and interoperability, less power consumption.[18]	Single user, Single application bound.

achieve server consolidation is lost from VC to bare metal [2][20]. Li et al. proved that the performance overhead depends more on the job to the job rather than feature by feature. Though VC is lightweight, it produced large performance overhead for certain jobs[3]. Containers and VMs are compared in a large scale data-centre environment along the dimensions of performance and SW development. Unlike VMs that have a strict limit on resources, VCs allows soft limits [4] [13] [14].

The performance difference in terms of throughput, response time and CPU utilization is evaluated between containers and VMs. Under certain scenarios, the Amazon AWS container performs worse than Amazon EC2 VMs [5][6][8]. Tosatto and Baresi et al. introduced micro cloud-a container-based solution that used TOSCA (Topology and Orchestration Specification for Cloud Applications) library for automatically adapting resource allocation[7][17]. Benchmark technique is introduced for the selection of cloud resources. The benchmark for containers called Doclite is used to evaluate the performance of containers and found to be 90% accurate than heavyweight VM benchmarks [9].

Docker had lower overhead than VM to execute NoSQL Cassandra [10]. As far as distributed systems are considered, for OS environments and servers like ngnixandRedis, docker swarm node consumes fewer resources and operational overheads than VMs [11]. Dhuraibi et al. introduced the vertical elasticity of Docker containers using autonomic computing. As the vertical elasticity is limited to the capacity of the host machine, live migration of container was experimented and proved as cost-effective resource utilization [12].

Docker is a Linux based container management system. The author elaborates the docker architecture and suggested how it can be adopted in windows environment [15]. A prototype for container-based cloud gaming system is built and its performance is benchmarked against KVM. The GPU takes advantage of containers than that of KVM. Though Docker proved as the best, its limitations on Windows, isolation, and security are yet to be overcome [16].

The architecture of Unikernel and its performance in the content delivery network is discussed. Madhavapeddy and Kuenzer et al. explained the advantage of using unikernel over containers in terms of security, isolation, speed and memory consumption [18][19].

The comparison among the various VT is listed in Table.1. It consolidates the pros and cons of each technology with a gist about their individual capabilities. Each technology outweighs the other in some criteria which concludes the significance of using the right technology to a suitable application.

4. Performance Measures

Although various cloud simulation tools are used nowadays, the evaluation and comparison of performance metrics under real-world scenarios can give a true picture of choosing appropriate technology for cloud services. This section studies the various performance evaluation methodologies, performance metrics, the workloads and benchmarks that are available for testing cloud services for different types of jobs.

In general, jobs can be classified as CPU intensive, memory intensive, disk intensive, network intensive, I/O intensive or transaction- intensive. The scientific applications are CPU intensive in which precision and performance are considered as utmost priority, hence termed as High Performance Computing (HPC) applications. Whereas, Commercial applications that solve business problems are disk and I/O intensive in which latency is of primary concern, hence termed as High Throughput computing (HTC).

4.1. Evaluation Methodologies

To understand the instantiation and execution of virtual resources in a real production site, it requires tools that simulate the real cloud environment. This section introduces the existing methodology tools that are used to evaluate performance measures.

The DoKnowMe (Domain Knowledge-driven methodology) is an abstract evaluation methodology with respect to the “class” in the object-oriented analysis and design [3]. By combining domain-specific facts DoKnowMe can be made more customized. For generating HTTP web service request to a target server and measuring its response time & throughput JMeter is used [5]. OpenNebula, OpenStack are the development and management tools for VMs and Docker Swarm for containers.

4.2. Workloads and Benchmarks

Iperf is a communication metric evaluating benchmark that consumes less resource and produces precise data throughput like STREAM which measures memory data throughput[3]. Bonnie++ measures storage transaction speed when the data is byte oriented and measures storage throughput if it is block sized data. The SpecJBB quantifies the CPU and Memory intensive jobs whereas the YSCB is a workload generator for data-intensive jobs to measure load, insert, read and update operations[4].

RUBiS is a renowned auction site of eBay that supports multi-tier web applications. PXZ benchmark is specifically designed for CPU measure, whereas Nuttcp and Netperf [2] are used to test network throughput and latency. SysBench and Linux binary copy (dd) [2] utility are powerful for Disk I/O evaluation. Blake2, 7-zip, OpenSSL (Open Secured Socket layer) are the benchmarks that are used to test the security level of the virtualization technologies[14].

4.3. Performance Analysis

The earlier works discussed the usage of the newest technologies in the development of cloud service. The current survey is a preliminary work to investigate the QoS measures of VM based and Container based technologies. The new technologies that

currently evolves can be classified as the variants of the two existing technologies VMs and Containers. Hence a brief comparative study of the two basic technologies can conclude to choose the right infrastructure for resource allocation in the cloud data center.

4.3.1. Deployment

Deployment refers to the launching of applications which is developed with management and orchestration frameworks. The memory occupied by the compute resources plays a vital role in deciding the launching time required [15]. As every VM has got its own guest OS, it consumes more time and memory to emulate than VCs. Kubernetes, the container framework comes with a monitoring tool that monitors the failed replicas and replaces them automatically, which is a property yet to be achieved in other technologies [15].

4.3.2. Degree of Interference

Isolation is the property measured by the degree of interference among applications/machines. In CPU intensive applications, the interference is alleviated by VMs because of the separate scheduler in the Guest OS. The shared OS kernel cause interruption leading to a denial of service attacks [4]. VM gives complete system isolation whereas the container is restricted to application isolation [3][15]. On the other hand, the containers inside VM achieve both the isolation of VM and speed of the container. For memory intensive applications, the containers and hosted VMs reacts more or less the same [4].

4.3.3. Auto Scaling and Elasticity

During resource provisioning, VMs are invoked or released through horizontal scaling (replication). With the advent of autonomic resource provisioning, a precise number of resources are planned to avoid over-commitment of resources [12]. The MAPE-K control loop is followed to automate the scaling process. For container based resources, though kubernetes and Docker swarm provides auto horizontal scaling, vertical elasticity is achieved by live container migration when there is a shortage of resources in the underlying host machine [12].

4.3.4. Soft & Hard limits

When application requires additional resources beyond their allocated limit, soft limit permits the applications to access the under-utilized resources of the host machine [4]. This enables better resource utilization in containers. But with VMs, soft limit is hard to achieve as the number of resources required are fixed during guest OS boot up which is referred as hard limits[4]. In the containers within VM technology, neighbor containers are adopted thus enabling elasticity of resources through soft limits [4].

4.4. Performance Metrics

The performance quality of the cloud service is the criteria for the selection of CSPs. The information to know how the cloud service responds throughout the service period helps to estimate the vendor selection. Hence the CSP opt for the best technology that can yield profit and customers. Generally, the following metrics are evaluated to choose the right virtualization technology.

4.4.1. Network and Security Performance

As far as network operations are concerned, containers suffer poorly with high packet rates. Lack of isolation at the system level in containers leads to poor security [14]. Hence nesting containers inside VMs are the solution to achieve high security in containers. Also, the neighbor containers within the same VM can be trusted [4]. Thus the security issues in the container based machines can be resolved.

4.4.2. Disk I/O Performance

Google Cloud SQL and Microsoft SQL Azure provide DBMS as a cloud service. In VC, the file system is shared and its journal batchesall the updates from multiple containers and commits as one transaction [1]. Hence each container waits for its update to get executed. This slows down the I/O operation and the shared file system causes interference between containers leading to isolation problem. On the other side, VM maintains its own journal of the separate file system which is the reason for its better performance. As such, VMs exhibits 86% of the increase in throughput, without compromising isolation, makes it better to host DBMS applications. The fact gets reversed when it comes to running distributed databases like Cassandra (NoSQL) in big data[10]. While running Cassandra in VM, the resource and the operational overheads of the virtualization layer affects the performance of application too. But, Dockerized Cassandra packs the application and their dependencies into Lightweight container thus consuming fewer resources and least overhead costs.

4.4.3. Migration Time and Cost

As VCs do not possess any Guest OS, they are Lightweight. This reduces memory size, cost and less time to migrate VCs [4] [8]. The lightweight VMs have a comparative benefit over their predecessors. The ultimate Unikernels that are still in its infancy for cloud services are the most economical and time-saving [18].

4.4.4. Latency & Throughput

For CPU intensive workloads like SpecJBB 2005 (supports 3-tier web application stack) containers are better. The YSCB benchmark for disk intensive applications (load/read/update operations) shows higher latency in VM, while unikernal applications outshine all other technologies [4]. Container and its extended technologies perform better than VM with respect to throughput.

4.4.5. CPU and Memory Performance

For High-Performance Computing (HPC) and High Throughput Computing (HTC) workloads, the container technology outshines the Virtual Machine [13]. Whereas, the memory intensive workloads are concerned VMs and container variants stands almost parallel with the better side on the VM [2]. But surprisingly, in deploying web service applications, VMs outperforms container variants by its performance metrics. The reason for this performance degradation is that the containers being embedded within VMs and not directly on bare metal.

As far as the distributed systems are concerned, Docker containers consume minimal memory, storage, CPU utility, boot time and power [11]. Hence containerization shows better performance and less operational overheads [6]. This is because of the large size of the OS image in the VMs that takes more time and memory to deploy. But this issue can be resolved by replacing traditional VMs with Lightweight Virtual Machines using smaller images like TinyCore, Lubuntu and Vector Linux [11].

In Unikernals [18], only the relevant runtime functionalities required to execute the application is embedded. Unikernels drastically minimizes the software size and makes it ultimate fast to deploy and execute. Hence Unikernel best suits for the upcoming era of micro-services [5] with its significance for interoperability and portability features.

A Brief comparison among the variants of VT under different parameter is discussed in Table.2. The individual data given under

each column varies depending upon the specifications like type of application, configuration of the underlying host machine, OS type and the type of hypervisor installed etc. The table infers the significance of using the suitable technology to an appropriate application.

Table2. Virtualization Technology specifications for various parameters

Virtualization Technologies / Parameters	VMs	VCs	Containers within VMs	Light weight VMs	Unikernels
Time to Instantiate[1] in Xen Hypervisor (in milliseconds)	6500	1711 Docker	200	431	31
***Image Size[1] (in MBs)**	913	61	53	3.7	2
***Throughput[1] (in MBs) Transmission, Receive**	23,24	45,43	52,44	29,25	50,33
Platform Dependency	Windows specific	Linux specific	Platform Independent	Platform Independent	Platform Independent
Number of VMs invoked per host	Dozen	In 100s	In 100s	Few 100s	In 1000s and more
Efficient Resource Utilization	Poor utilization	Better utilization	Better utilization	Better utilization	Best utilization
Isolation Level Achieved	System level	Application level	System & application level	System & application level	System & application level
Application Portability	Hypervisor Dependent	OS dependent	OS dependent	Kernel specific	Kernel Specific
Security Level	Highest security come with cost	Poor security	Medium security	High security	High security
Profit at CSP point of view	Incurs more cost, not Profitable	Profit oriented	Nominal profit	High profit	More profit oriented
Orchestration frameworks	Open Nebula	Docker Swarm	Open Stack	Kubernetes	Unik, Jitsu[2]
Application Suitability	Security and Isolation intensive applications	Disk I/O Intensive applications	CPU, Intensive applications	Memory Intensive applications	Micro services
Consumer choice for VT selection	Incurs more cost	Incurs Less cost	Incurs Less cost	Incurs Less cost	Cost saving
Examples	Xen, KVM	Docker	vSphere[3]	Tinyx, Clear Linux	ClickOS, IncludeOS[4]

* All tests are run on an x86--- 64 server with an Intel Xeon E5--1630 v33.7GHz CPU (4 cores) and 32GB RAM. An Image is a virtual disk which has the operating system, application libraries, application code and configuration, etc. The image size depends on the application code and its required libraries. The number of machines that can be invoked on a physical host depends on the resource capacity of the underlying host machine.

[1]https://datatracker.ietf.org/meeting/96/materials/slides-96-nfvrg-3

[2]https://github.com/cetic/Unikernels

[3]https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/whitepaper/vmw-wp-container-on-vms-a4-final-web.pdf

[4]https://www.duo.uio.no/bitstream/handle/10852/63618/1/thomasoddsund.pdf

5. Conclusion and Future Perspectives

The performance methodologies, metrics, benchmarks and job types that are brought into focus from the survey of several works helps researchers to choose the right type of resource for cloud resource provisioning. The current study shows that the ultimate purpose of the virtualization technology is to provide a cost-effective solution to the CSPs and quality of service to the end users with less price. Although the IT industry that works behind this goal, adopts the newest technologies, each technology differs depending upon the job types and the application requests.

From the earlier literature works, it is clear that neither of the virtualization technologies nor its variants can standalone dominant for all types of cloud services. The technology to be chosen varies depending upon whether it is web service, data analytics, video streaming, scientific or commercial applications.

Selection of the right virtualization technology favors efficient resource utilization in the cloud. It is evident that corporate giants like Google, Microsoft, Amazon, and others employ either an individual technology or the combination of virtualization technologies depending upon the performance requirements of the cloud applications. Hopefully, this review may help the researchers to narrow down to the right technology for their future work in efficient resource utilization.

6. References

[1] A. A. A. Mardan and K. Kono, "Containers or Hypervisors: Which Is Better for Database Consolidation?," in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Luxembourg, Luxembourg, 2016, pp. 564–571.

[2] C. G. Kominos, N. Seyvet, and K. Vandikas, "Bare-metal, virtual machines and containers in OpenStack," in *2017 20th Conference on Innovations in Clouds, Internet and* Networks *(ICIN)*, Paris, 2017, pp. 36–43.

[3] Z. Li, M. Kihl, Q. Lu, and J. A. Andersson, "Performance Overhead Comparison between Hypervisor and Container Based Virtualization," in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, Taipei, Taiwan, 2017, pp. 955–962.

[4] P. Sharma, L. Chaufournier, P. Shenoy, and Y. C. Tay, "Containers and Virtual Machines at Scale: A Comparative Study," in *Proceedings of the 17th International Middleware Conference on - Middleware '16*, Trento, Italy, 2016, pp. 1–13.

[5] T. Salah, M. J. Zemerly, C. Y. Yeun, M. Al-Qutayri, and Y. Al-Hammadi, "Performance comparison between container-based and VM-based services," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, Paris, 2017, pp. 185–190.

[6] K. Kumar and M. Kurhekar, "Economically Efficient Virtualization over Cloud Using Docker Containers," in *2016 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, Bangalore, India, 2016, pp. 95–100.

[7] A. Tosatto, P. Ruiu, and A. Attanasio, "Container-Based Orchestration in Cloud: State of the Art and Challenges," in *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, Santa Catarina, Brazil, 2015, pp. 70–75.

[8] A. M. Joy, "Performance comparison between Linux containers and virtual machines," in *2015 International Conference on Advances in Computer Engineering and Applications*, Ghaziabad, India, 2015, pp. 342–346.

[9] B. Varghese, L. T. Subba, L. Thai, and A. Barker, "Container-Based Cloud Virtual Machine Benchmarking," in *2016 IEEE International Conference on Cloud Engineering (IC2E)*, Berlin, Germany, 2016, pp. 192–201.

[10] S. S. Emiliano Casalicchio and Lars Lundberg, "Performance Evaluation of container and virtual machine running cassandra workload.," *IEEE*.

[11] N. Naik, "Migrating from Virtualization to Dockerization in the Cloud: Simulation and Evaluation of Distributed Systems," in *2016 IEEE 10th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments (MESOCA)*, Raleigh, NC, USA, 2016, pp. 1–8.

[12] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Autonomic Vertical Elasticity of Docker Containers with ELASTICDOCKER," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, Honolulu, CA, USA, 2017, pp. 472–479.

[13] P. R. Desai, "A Survey of Performance Comparison between Virtual Machines and Containers," *Int. J. Comput. Sci. Eng.*, vol. 4, p. 6, 2016.

[14] R. K. Barik, R. K. Lenka, K. R. Rao, and D. Ghose, "Performance analysis of virtual machines and containers in cloud computing," in *2016 International Conference on Computing, Communication and Automation (ICCCA)*, Greater Noida, India, 2016, pp. 1204–1210.

[15] S. Singh and N. Singh, "Containers & Docker: Emerging roles & future of Cloud technology," in *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, Bangalore, India, 2016, pp. 804–807.

[16] T. Kamarainen, Y. Shan, M. Siekkinen, and A. Yla-Jaaski, "Virtual machines vs. containers in cloud gaming systems," in *2015 International Workshop on Network and Systems Support for Games (NetGames)*, Zagreb, Croatia, 2015, pp. 1–6.

[17] L. Baresi, S. Guinea, G. Quattrocchi, and D. A. Tamburri, "MicroCloud: A Container-Based Solution for Efficient Resource Management in the Cloud," in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, New York, NY, USA, 2016, pp. 218–223.

[18] Anil Madhavapeddy, Richard Mortier1, Charalampos Rotsos, David Scott2, Balraj Singh, Thomas Gazagnaire3, Steven Smith, Steven Hand and Jon Crowcroft, "Unikernels: Library Operating Systems for the Cloud" in *ASPLOS'13, March 16–20, 2013, Houston, Texas, USA.* Copyright 2013 ACM 978-1-4503-1870-9/13/03.

[19] Simon Kuenzer, Anton Ivanov, Filipe Manco, Jose Mendes, Yuri Volchkov, Florian Schmidt, Kenichi Yasukata, Michio Honda, Felipe Huici "Unikernels Everywhere: The Case for Elastic CDNs" in VEE '17 April 08-09, 2017, Xi'an, China 2017 Copyright held by the owner/author(s). ACM ISBN 978-1-4503-4948-2/17/04.

[20] H.A.Duran-Limon, M.Siller, G.S.Blair, A.Lopez, J.F.Lombera-Landa, "Using lightweight virtual machines to achieve resource adaptation in middleware" *in IET Softw.*, 2011, Vol. 5, Iss. 2, pp. 229–237.