ISSN (e): 2250 – 3005 || Volume, 07 || Issue, 8|| August – 2017 ||
International Journal of Computational Engineering Research (IJCER)

Container-Based Ml Services in Cloud Environment to Improve Network Performance & Deployment

Amiya Kumar Sahoo [1] Amit Kumar Jha [2] Ashis Acharya[3] Rakhi Jha[4]
Department of Computer Science & Engineering, Aryan Institute of Engineering & Technology, Bhubaneswar
Department of Computer Science & Engineering, Raajdhani Engineering College, Bhubaneswar
Department of Computer Science & Engineering, Capital Engineering College, Bhubaneswar
Department of Computer Science & Engineering, NM Institute of Engineering & Technology, Bhubaneswar

ABSTRACT: *Cloud computing has an emerged as an important relevence to improve resource utilization, efficiency, flexibility, However, cloud platforms cause performance degradations due to their virtualization layer and may not be appropriate for the requirements of high performance applications, such as machine learning. The Research tackles the problem of improving network performance in container based cloud instances to create a viable alternative to run network intensive machine learning applications. Our approach consists of deploying container based machine learning based apps via LXC (Linux Container) cloud instances to increase the available bandwidth & performance. so to evaluate the efficiency of this approach and the overhead added by the container- based cloud environment, we ran a set of experiments to measure throughput, latency, bandwidth utilization, and completion times. The outcomes proves that this approach adds minimum overhead in cloud environment as well as increases throughput and reduces latency. Containers are a lightweight virtualization solution to replace virtual machines for deploying cloud applications as they are less resource and time consuming. Easy deployment of applications on containers is done using kubernetes which helps user to analyze applications with various network topologies improving transfer rate of large data sets which are required for machine learning models*

KEYWORDS: *Cloud Computing Containers, virtualization machine learning, Network Performance, Link Aggregation, Container-Based Cloud, Container Network, Performance*

I. INTRODUCTION

Today, cloud computing is rapidly developing and large scale application. As a kind of network-based computing, cloud computing can provide shared software and hardware resource to users by using network, which realizes the reasonable distribution of information and resource. Meanwhile, cloud computing brings along the changes of the traditional data center, which produces a new generation of data center: cloud data center. By building computing resources, storage resources and network resources into dynamic virtual resource pool using virtualization technology But now cloud data center faces with many problems such as low resource utilization, application and platform can not be decoupled, application runtime environment limitations are strong, and operational staff control decrease, which limit the development of it. Virtual machines (VMs) are an infrastructure as a service (IaaS) focusing on hardware virtualization whose techniques have been used at the infrastructure layer. To achieve sharing and elasticity of resources, the cloud makes use of such virtualization techniques for administering scheduling, provisioning and security. However, VMs suffer from slow start up time and exhibit lower densities even when full. Applications occupy entire host operating system (OS) in VMs and there are also several limitations on storage and Containers offer as a lightweight virtualization solution to deploying applications across various domains and sectors. They allow infrastructure and platform to be shared in a secure and portable manner, along with application packaging and management. They facilitate faster deployment of applications and have faster start up times. They are less resource and time consuming and can be scaled up or down providing higher density levels than full VMs. They facilitate easier and portable across infrastructure deployment of applications in an interoperable way. Using containers will accelerate agile application development of distributed applications providing an additional layer of protection by isolating applications and the host, without using incremental resources. They also allow easy updates to applications. Fig. 1 shows the difference between traditional hypervisor and container-based architectures

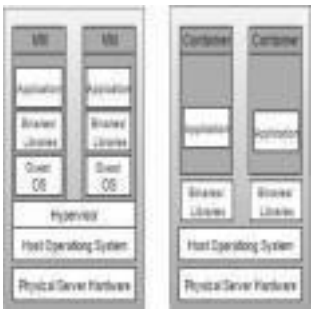

Fig.1: Traditional hypervisor architecture on the left and a container-based architecture on right

Container technology is a virtualization technology at the operating system level, it can provide a separate file system, network and process context space for each running server without modifying the host operating environment. So that, each service can arbitrarily change the contents of the operating system files, configuration of network and user in their own virtual running space, without damage to real host system files. Although an instance of the operating system is shared between containers, they run independently and without interference. Besides, do not relaying on the integrity of the operating system makes container more lightweight and fast start than virtual machine, which meets the needs of cloud data center. Cloud computing infrastructures support rapid resource provisioning, which in turn is a suitable option to deploy a container environment. Primarily due to the advances of cloud platforms, there are possibilities to develop new business models based on the pay-per-use billing structure. Also, there are cloud computing technologies (specifically those focused on network and storage) that provide lower performance degradation. However, the use of cloud environments for processing distributed applications has traditionally been avoided when it requires high bandwidth and throughput as well as low latency. Traditional cloud providers, which use virtualization technologies that are not optimized for the execution of distributed applications, add significant overheads. Our main challenge in this paper is to improve network performance in the container-based cloud environment for applications. We chose to increase the network performance by using the IEEE 802.3ad link aggregation standard. It gives and uses a standard method to combine multiple physical links that can be used as a single logical link. The standard is a layer 2 control protocol that can be used to automatically detect, configure, and manage a single logical link with multiple physical links between two adjacent enabled devices. Thus, link aggregation provides higher availability and capacity while network performance improvements are obtained using existing hardware (IEEE 802.3ad requires support in a network switch). Also, our goal is to target a high- performance cloud environment by deploying container- based instances, where applications may run. We chose container-based technologies to enable multiple isolated Linux systems to run on a single host through Namespaces (providing isolated user environments in the form of containers) and cgroups (providing resource management and accounting). LXC (Linux Container) technology is an important part of this cloud infrastructure, because it is a free software that provides a powerful set of user space tools and utilities to manage Linux containers

II. RELATED WORK

In this section, we provide a brief background about machines placement, Docker containers, Docker Container Networking. We also described related work reported in the literature with regards to performance of Docker containers when compared to the performance of VMs. ALSO we should focus on container network models. There are two mainstream models called Container Network Model (CNM) and Container Network Interface (CNI).

A. Virtual machine placement:

Nowadays, cloud computing has been widely used.The Virtual Machines (VMs) are created on servers in cloud computing. With the rapid development of Cloud computing technologies, the management of Cloud resources has become crucial to Cloud Service Providers (CSPs). CSPs provide on demand services to the end users through provisioning Virtual Machines (VMs) in their available Physical Machines (PM)s. In order to reduce the Total Cost of Ownership(TCO), CSPs consolidate multiple VMs in each of the available PMs. In this context, significant number of research work have already been done which indicate the benefit of running multiple VMs in a PM. The VM scheduling on servers for energy saving in the cloud computing has been studied. The Virtual Machine Scheduling Algorithm (VSA) is proposed to schedule VMs in cluster environments. However, it is not effective and has high time complexity. The VM scheduling on servers for energy saving inthe cloud computing has been studied. The Virtual Machine Scheduling Algorithm (VSA) is we propose a new scheduling method called Energy-aware Virtual Machine Placement (EVP)method to schedule VMs that can reduce power consumption complexity. Power consumption of a PM depends on the number of

VMs deployed, reserved resources of VMs, and the state ofthe processes running within the VMs. There are significant number of works that try to minimize the the overall power consumption of a data- center by reducing the number of active PMs. They classify this problem of mapping the VMs to minimum number of PMs satisfying the constraints of physical resources and the requirements of the VMs as the VM Placement Problem

B. Docker Container

Docker containers are lightweight, highly portable and scalable. Such features become attractive to develop containerized services (or microservices). In the literature, container-based services are reported to always outperform VM-based services. For example, it was shown in that containers outperform virtual machines in terms of execution time, latency, throughput, power consumption, CPU utilization and memory usage. However, according to Amazon AWS documentation. it was reported that Docker containers are deployed on top of virtual machines (VMs), and not on bare-metal hardware. This contradicts the common practice of container deployment which is widely adopted in the literature of deploying container on bare-metal hardware. This was the primary motivation for our research work in which we aim to investigate and assess the performance difference resulting from deploying services using VMs (virtual machines) and using Docker containers.

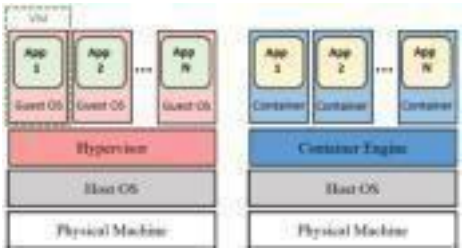

Fig.A&B: Common approach of hosting of VMs vs. Containers

The Services deployed using containers are expected to take less execution time thus resulting in less latency over virtual machines as in. In addition, containers were shown to introduce lower power consumption over virtualized technologies.

C. Docker Container Networking:

As an important part of container technology, container network mainly solves the problem of communication between containers in the case of ensure container isolation. Since container is a new technology, container network is not yet mature, and many container network solutions were proposed to solve the problems of container network. Each solution has its own characteristics, and how to choose appropriate solution according to different scenarios is not only very important for the efficiency of communication between containers but also of great significance in improving cloud data center performance. In order to provide experiences and references for choosing and deploying appropriate container network, we introduce three kinds of mainstream container network solutions (Flannel, Docker Swarm Overlay, Calico), and design experiments to measure and evaluate the performance of them

Flannel

Flannel is a network planning service designed for Kubernetes. The deployment and configuration for Flannel is relatively simple, through the Etcd assigned IP address, it can manage multiple containers cross host. Flannel is designed to be lightweight and superior in performance. It can solve the IP conflict in Docker default configuration by replanning the allocation of IP address. When communication, Flannel will assign each Docker daemon an ip segment Through Etcd to maintain a cross host routing table, the IP between containers can be connected to each other. When two containers across the host communicate with each other, they will modify the destination address and source address of the packet, and unpack it after sending to the target host through routing table. Flannel is the most mature in container network solutions, but since each host is a separate subnet network architecture, it is not possible to achieve fixed IP container drift and lack of flexibility. Beside, because of the lack of multi-subnet isolation mechanism, high dependency on the upper design and other defects, Flannel can only be used for the container cluster which have low requirement for flexibility

Docker Swarm Overlay

Docker Swarm Overlay is a network solution that Docker uses the Swarm framework to solve network communication problems between containers. Swarm Overlay was originally native to Docker, the deployment and configuration is the simplest. previously, Overlay needed to add additional key- value storage (Consul, Etcd

etc) in the Swarm cluster to synchronize the network configuration to ensure that all containers were in the same network segment. This storage was built in Docker and integrated the support of Overlay Networks, which simplifies the complexity of network configuration. Overlay integrates load balancing and service address function through the Swarm, it provides a DNS address for each service and maintains a common port. Besides, it implements the monitoring and management mechanism to maintain the operation of service state. However, Overlay solution uses Vxlan for cross host communication and requires encapsulation and decapsulation, which lose the performance of network, so it is just suitable for the construction of a simple network with low performance.

Calico

Calico is a high performance data center solution base on BGP protocol, implemented through three layer routing. By compressing the entire Internet's extensible IP network principles to the data center level, Calico uses vRouter for data forwarding at each compute node and spreads the workload routing information to the entire Calico network through the BGP protocol to ensure that all data traffic are completed through IP packet. Similar to the general routing scheme, Calico runs a large number of routing tables on the host, these routing tables are dynamically generated and managed by Calico through its own components, with no involvement of tunnels or NATs and less performance loss. These make sure that Calico is of high performance.The Calico solution is suitable for networks that require high performance and high isolation. Each container in Calico has its own network protocol stack, which facilitates node interconnection. Because Calico directly uses the data center network structure, when deploying network, there is no need to rely on independent network equipment, so the transfer efficiency higher.

Container Network Model

CNM is proposed by Docker's Libnetwork project, which focuses on container networks research, providing standardized interfaces and components between the Docker daemons and network drivers. architecture is depicted in Figure below. CNM mainly includes three components named sandbox, endpoint and network. Sandbox is an isolated network operating environments that preserves the configuration of container network stack, End point represents the point which container access to network, we can think of an endpoint as a physical network card. Network represents a set of endpoints that can communicate directly to each other, which based on Linux bridge or vlan.The emergence of CNM makes it possible that different containers within the same subnet can run on different hosts. LibNetwork can meet the needs of the upper and lower levels. For the upper application, Docker daemon can perform the creation and management of network by calling the APIs providing byLibNetwork; for the lower application, the five drivers built into LibNetwork can provide different types of network services. CNM occupies the core of LibNetwork, which is the key to provide network services.

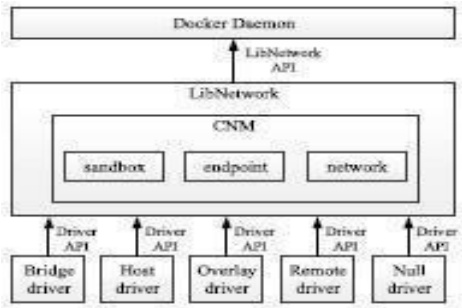

Figure d. CNM Architecture.

Container Network Interface(CNI)

CNI is a container network specification with compact structure proposed by CoreOS. The key to CNI lies in the simple contract between the container Runtime and the network plugin. CNI provides a universal container network model, container can be added to multiple networks driven by different plugin, and each network has its own corresponding plugin and unique name. Therefore, CNI can be compatible with other container technology and the upper system. CNI provides a universal container network model, container can be added to multiple networks driven by different plugin, and each network has its own corresponding plugin and unique name. Therefore, CNI can be compatible with other container technology and the upper system.CNM and CNI are not two completely contradictory models, they can be transformed into each other. Comparing Figure 2.3 to Figure 2.4, Sandbox in CNM is consistent with Container Runtime in CNI. The Endpoint in CNM is implied in the ADD/DELETE operation in CNI.The CNI plugin only needs to provide two commands throughout the model: one for adding network interfaces to the specified network and the other to remove it.

These two commands are called when container is created and destroyed. CNI architecture is depicted in Figure e.

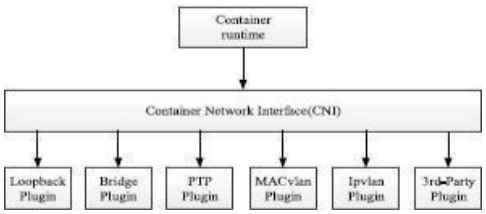

Figure e. CNI Architecture.

Introducing IEEE 802.3ad
it is link aggregation enables you to group Ethernet interfaces at the physical layer to form a single link layer interface, also known as a link aggregation group (LAG) or bundle. The IEEE 802.3ad enables dynamic link aggregation and dis- aggregation by exchanging the packets in Docker cluster. In this case, the switch equipment dynamically groups similar ports into a single logical link, increasing the bandwidth and balancing the load for the Containers. Some users needed more performance in their network than a single Fast Ethernet link can provide, but cannot afford the expense or do not need the bandwidth of a higher-speed Gigabit Ethernet link. Using IEEE 802.3ad link aggregation in this condition provides increased port density and bandwidth at a lower cost. For example, if you need 450 Mbps of bandwidth to transmit data and have only a 100-Mbps Fast Ethernet link, creating a LAG bundle containin five 100-Mbps Fast Ethernet links is more cost-effective than buying a single Gigabit Ethernet link.

Kubernetes
Kubernetes provides the means to support container-based deployment within Platform-as-a-Service (PaaS) clouds, focusing specically on cluster-based systems. It allows to de- ploy multiple\pods"across physical machines, enabling scale out of an application with dynamically changing workload. Each pod can support multiple Docker containers, which are able to make use of services (e.g. file system and I/O) associated with a pod. With significant interest in supporting cloud native applications (CNA). One of the most popular solutions for container-based virtualization is using Docker for container packaging with Kubernetes for multihost container management. Kubernetes follows the master slave model, which uses a master to manage Docker containers across multiple Kubernetes nodes (which are physical or virtual machines). A master and its controlled

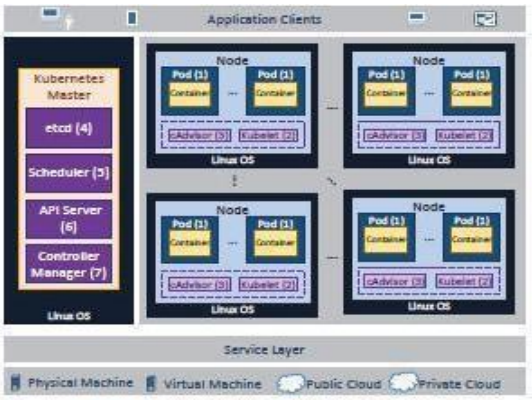

Fig f. System Architecture

etcd: As a storage component, etcd is used to store the state of the system, allowing the other master components to synchronize themselves to the desired state by watching etcd Scheduler. The scheduler is responsible for scheduling each pod on a node in the system
API Server. The API server is responsible for receiving commands and manipulating the data for Kubernetes objects (such as pods) accordingly in the system. User scan send commands to the API server by using the Kubernetes command line interface (CLI), kubectl
Controller Manager. The controller manager monitors etcd and regulates the state of the entire system. In other words, if the state of the system changes, the controller manager will force the system into the desired state using the Kubernetes API.

Prometheus

Prometheus is an open-source systems performance monitoring and alerting toolkit originally built at SoundCloud. Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community. It is now a standalone open source project and maintained independently of any company Monitoring nodes constitute a –Kubernetes cluster.| A developer can deploy an application in the docker containers via the assistance of the Kubernetes master. Typically, an application is divided into one or more tasks executed in one or more containers master is the central controller of the system. It contains four Kubernetes master components—etcd, Scheduler, APIServer, and Controller Manager—which are used to controland manage the nodes and the containers in the system. Briefly,the functions of the four master components in our system areas follows:· applications & application servers is an important part of the today's DevOps culture & process. You want to continuously monitor your applications and servers for application exceptions, server CPU & memory usage, or storage spikes. You also want to get some type of notification if CPU or memory usage goes up for a certain period of time or a service of your application stops responding so you can perform appropriate actions against those failures or exceptions. With monitoring, there are a number tools out there such Amazon CloudWatch, Nagios, New Relic, Prometheus and others.

Prometheus Component

a. Promtheus Server:

Prometheus has a main central component called Prometheus Server. As a monitoring service, Prometheus servers monitor a particular thing. That thing could be anything: it could be a an entire Linux server, a stand-alone Apache server, a single process, a database service or some other system unit that you want to monitor. In Prometheus terms, we call the main monitoring service the Prometheus Server and the things that Prometheus monitors are called Targets. So the Prometheus server monitors Targets. As said earlier, "Targets" can also refer to an array of things. It could be a single server or a targets for probing of endpoints over HTTP, HTTPS, DNS, TCP and ICMP (*Black-Box Exporter), or it could be a simple HTTP endpoint that an application exposes through which the Prometheus server gets the application's health status from. Each unit of a target such as current CPU status, memory usage (in case of a Linux server Prometheus Target) or any other specific unit that you would like to monitor is called a metric. So Prometheus server collects metrics from targets (over HTTP), stores them locally or remotely and displays them back in the Prometheus server.

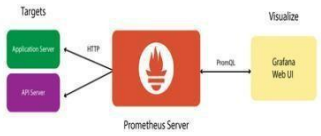

Fig h (a).Prometheus Server

You get the metrics information by querying from the Prometheus's time-series database where the Prometheus stores metrics and you use a query language called PromQL in the Prometheus server to query metrics about the targets. In different words, you ask the Prometheus server via PromQL to show us the status of a particular target at a given time.

b. Visualization Layer with Grafana:

You use Grafana, the visualization layer, as the its a compononet of prometheus server to visualize metrics stored in the Prometheus time-series database. So, Instead of writing PromQL queries directly into the Prometheus server, you use Grafana UI boards to query metrics from Prometheus server and visualize them in the Grafana Dashboard.

c. Alert Management with Prometheus Alert Manager Prometheus also has a Alert Management component called AlertManager for firing alerts via Email or Slack or other notification clients. You define the Alert Rules in a file called alert.rules through which the Prometheus server reads the alert configurations and fires alerts at appropriate times via the Alert Manager component. For example, if the Prometheus server finds the value of a metric greater than the threshold that you defined in the alert.rules file for more than 30 seconds, it will trigger the Alert Manager to fire an alert about the threshold and the metric. We will see how AlertManager works with Prometheus and how do we setup in the Prometheus stack in some later post. The above 3 components are the basis of the entire Prometheus Monitoring system. You need the central Prometheus server, a target and a visualization layer.

III. LITERATURE REVIEW

This section describes the primary related works aimed at improving the network performance and deploying Container Based instances or Services. Our first scenario consists of evaluating network performance and then second scenario consist deploying Container based ML services The Authors Wattanasomboon, P., & Somchit, Y. (2015, October)[1] proposes the method called EVP(Energy-Aware virtual machine placement) to method to schedule VMs that can reduce power consumption. We also formulate power consumption model to evaluate the performance of the EVP method. Finally, we evaluate the EVP method by simulation. On The basis of experimental results show that the EVP method has better performance. There are many researches of VM scheduling for energy saving. Befor using EVP method for our approach, we have to check performance of our VMs and containers and choose which platform is better to to deploy the ML(machine learning)Services.the Salah, Tasneem, et al.[5] stated the performace comparison between VMs and Containers and gave theirs results as Container always outperforms the VM in terms of the parameters like Cpu Utilization, latency, Throughput, Memory usage, Execution Time etc. So Services deployed using containers are expected to take less execution time thus resulting in less latency over virtual machines as in.

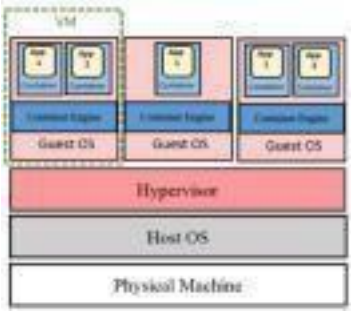

Fig3. Hosting of Containers in Amazon cloud Furthermore, according to Amazon , Docker containers are deployed on top of virtual machine, which is against the common practice of deployment as depicted in Fig.A&B and Fig.3 shows the container deployment approach adopted by Amazon cloud. The Author Zeng, Hao, et al. IEEE, 2017,[8] stated Container technology As an important part of container technology, container network mainly solves the problem of communication between containers in the case of ensure container isolation. Since container is a new technology, container network is not yet mature, and many container network solutions were proposed to solve the problems of container network. Each solution has its own characteristics, and how to choose appropriate solution according to different scenarios is not only very important for the efficiency of communication between containers but also of great significance in improving cloud data center performance. In order to provide experiences and references for choosing and deploying appropriate container network, we introduce three kinds of mainstream container network solutions (Flannel, Docker Swarm Overlay, Calico), and design experiments to measure and evaluate the performance of them. After Forming or Constructing the network for container we wanna improve the performance of the network. So the Author Rista, Cassiano, et al. IEEE, 2017.[6] published a paper which talks about the network performace improvement, so auther aims at provide a suitable approach to deploy dynamic link configuration and network link aggregation by using IEEE 802.3ad. so we used the std.802.3ad for improving the performance of our approach, so our goal is to target a high-performance cloud environment by deploying container-based instances or services. Finally the auther Medel, Víctor, et al. IEEE,2017,[10] provides the means to support container based deployment within Platform-as-a-Service(PaaS) clouds, focusing specifically on cluster-based systems. Kubernetes enables deployment of multiple pods" across physical machines, enabling scale out of an application with dynamically changing workload., Kubernetes provides a useful approach to achieve this. One of the key requirements for CNA is support for scalability and resilience of the deployed application, making more effective use of on-demand provisioning and elasticity of cloud platforms. Containers provide the most appropriate mechanism for CNA, enabling rapid spawning and termination compared to Virtual Machines (VMs). Kubernetes follows the master- slave model,which uses a master to manage Docker containers across multiple Kubernetes nodes (which are physical or virtual machines). A master and its controlled nodes constitute a –Kubernetes cluster.| A developer can deploy an application in the docker containers via the assistance of the Kubernetes master. So for deployement purpose we prefer the kubernetes to deploy ML services or instances.

IV. FUTURE WORK

We will do the performance comparison on PMs, VMs, Containers, kubernetes ISTIO and also Evaluate the ML services on this plateforms as this is our primary concern. we trying to the monitor performance of containers based ml services by using the prometheus which is a open-source monitoring and altering toolkit.

REFERENCES

[1]. Ernst & Young(1996),-Keeping Electronic Records Forever‖, Records Wattanasomboon, Pragan, andYuthapong Somchit. "Virtual machine placement method for energy saving in cloud computing." Information Technology and Electrical Engineering (ICITEE), 2015 7th International Conference on. IEEE, 2015.

[2]. Su, Nan, et al. "Research on virtual machine placement in the cloud based on improved simulated annealing algorithm." World Automation Congress (WAC), 2016. IEEE, 2016.

[3]. Karmakar, Kamalesh, Sunirmal Khatua, and Rajib

[4]. K. Das. "Efficient virtual machine placement in cloud environment." Advances in Computing, Communications and Informatics (ICACCI), 2017 International Conference on. IEEE, 2017.

[5]. Yang, Song, et al. "Reliable virtual machine placement and routing in clouds." arXiv preprint arXiv:1701.06005 (2017).

[6]. Salah, Tasneem, et al. "Performance comparison between container-based and VM-based services." Innovations in Clouds, Internet and Networks (ICIN), 2017 20th Conference on. IEEE, 2017.

[7]. Rista, Cassiano, et al. "Improving the Network Performance of a Container-Based Cloud Environment for Distributed Systems." High Performance Computing & Simulation (HPCS), 2017 International Conference on. IEEE, 2017.

[8]. Xu, Pengfei, Shaohuai Shi, and Xiaowen Chu. "Performance Evaluation of Deep Learning Tools in Docker Containers." Big Data Computing and Communications (BIGCOM), 2017 3rd International Conference on. IEEE, 2017.

[9]. Zeng, Hao, et al. "Measurement and Evaluation for Docker Container Networking." Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2017 International Conference on. IEEE, 2017

[10]. Mao, Ying, et al. "Draps: Dynamic and resource- aware placement scheme for docker containers in a heterogeneous cluster." Performance Computing and Communications Conference (IPCCC), 2017 IEEE 36th International. IEEE, 2017

[11]. Model, Víctor, et al. "Modelling performance & resource management in kubernetes." 2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC). IEEE, 2016.

[12]. Model. Víctor, et al. "Adaptive application scheduling under interference in kubernetes." Utility and Cloud Computing (UCC), 2016 IEEE/ACM 9th International Conference on. IEEE, 2016

[13]. Tsai, Pei-Hsuan, et al. "Distributed analytics in fog computing platforms using Tensorflow and Kubernetes." Network Operations and Management Symposium (APNOMS), 2017 19th Asia-Pacific. IEEE, 2017.

[14]. Chang, Chia-Chen, et al. "A Kubernetes-Based Monitoring Platform for Dynamic Cloud Resource Provisioning." GLOBECOM 2017-2017 IEEE Global Communications Conference. IEEE, 2017

[15]. Xie, Xiao-Lan, Peng Wang, and Qi Wang. "The performance analysis of Docker and rkt based on Kubernetes." 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD). IEEE, 2017

[16]. Rovnyagin, Mikhail M., et al. "Using the ML-based architecture for adaptive containerized system creation." Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018 IEEE Conference of Russian. IEEE, 2018