How cloud infrastructure can be built with low resource across multiple cloud platforms?

Karalyne Khongbri

Submitted as part of the requirements for the degree
of MSc in Cloud Computing
at the School of Computing,
National College of Ireland
Dublin, Ireland.

May 2016

Supervisor Dr. Ralf Bierig

Abstract

At present, Cloud Computing (CC) has been leading platform over the Internet. Virtualization is one significant part for the growth of cloud computing and to manage infrastructure across multi-cloud vendors. In this research, we study about Docker container and address to achieve high availability by building virtualizaation in cloud infrastructure deployment in multi-cloud framework and to over come the networking issues face by Docker. We focused to manage cloud infrastructure management with less resources and improve the networking server by creating private DNS server which will connect to all the container deploy in different cloud platform. The novel approach we brought, in this research is improvement of the cloud infrastructure management in multi cloud platform. This study is helpful to solve the networking issues in Docker framework. We used private DNS, as a networking server each cloud platform.This it result to improve connectivity among the containers deploy in different cloud vendor. We deploy multiple number of containers which is then connecting to the DNS server. To check the test performance we used YCSB tool. With the implementation of our proposed architecture, results to better performance of the MongoDB. We also test our proposed infrastructure by implementing in three different cloud vendor which prove the best performance of our infrastructure. We achieved, high availability over MongoDB clustered infrastructure when implement and compare among three different cloud platform.

Key Points- Virtualization, Docker container, MongoDB, YCSB, Live Migration, Cloud Infrastructure Management.

Submission of Thesis to Norma Smurfit Library, National College of Ireland

Student name: Karalyne Khongbri. Student number: X14125081

School: School of Computing Course: MSc in Cloud Computing

Degree to be awarded: MSc in Cloud Computing

Title of Thesis: How Cloud Infrastructure Can Be Built with Lower Resources across Multiple Cloud Platforms?

One hard bound copy of your thesis will be lodged in the Norma Smurfit Library and will be available for consultation. The electronic copy will be accessible in TRAP (http://trap.ncirl.ie/), the National College of Ireland's Institutional Repository. In accordance with normal academic library practice all theses lodged in the National College of Ireland Institutional Repository (TRAP) are made available on open access.

I agree to a hard bound copy of my thesis being available for consultation in the library. I also agree to an electronic copy of my thesis being made publicly available on the National College of Ireland's Institutional Repository TRAP.

Signature of Candidate: Karalyne Khongbri

For completion by the School:
The aforementioned thesis was received by_____ Date: 30/05/2016

This signed form must be appended to all hard bound and electronic copies of your thesis submitted to your school

Submission of Thesis and Dissertation

National College of Ireland

Research Students Declaration Form

(Thesis/Author Declaration Form)

Name: Miss.Karalyne Khongbri Student Number: X14125081

Degree for which thesis is submitted: MSc in Cloud Computing

Material submitted for award

(a) I declare that the work has been composed by myself.

(b) I declare that all verbatim extracts contained in the thesis have been distinguished by quotation marks and the sources of information specifically acknowledged.

(c) My thesis will be included in electronic format in the College Institutional Repository TRAP (thesis reports and projects)

(d) ***Either*** *I declare that no material contained in the thesis has been used in any other submission for an academic award.

Or *I declare that the following material contained in the thesis formed part of a submission for the award of

Master of Science in cloud computing awarded by QQI at level 9 on the National Framework of Qualification

(State the award and the awarding body and list the material below)

Signature of research student:Karalyne Khongbri.

Date: 30th May 2016.

Acknowledgement

Firstly, I would like to express my sincere gratitude to my supervisor Dr. Ralf Beirig for his great support, he had given me during the short period to successfully complete my dissertation. I would also like to thank my guide for his patience towards me and for sharing immense knowledge. His motivation has helped me to successfully complete my research work and the thesis in every possible ways. In each meeting with him, he has always encouraged me to do the best in all ways and his advice has increase my confidence to keenly work hard for this study. I am very much thankful for the efforts he had put to helped me and especially his valuable time he had given me for the meeting whenever I request for the appointment. He had always given me the positive aspect of my work and implementation. My sincere thanks also goes to our Head Of Cloud Computing Dr. Horacio Gonzalez-Velez who had always support and guide me in the completion of my course. I would also like to thanks Robert Duncan, IT support department and all the NCI staff for the help and their valuable time they had provided in completion of my course. I would like to thank for the NCI on-line resources which help me to improve my writing and referencing skills. I am thank full to all the support given by everyone all the time directly and indirectly throughout this year of MSc Cloud Computing.

I would like to thank all my classmates and all my friends, who had always been there for me to provide help, motivation, advice and great support in difficult time. Their motivation and advice has given me more confidence to concentrate on my work and to successfully complete my course.

I express my sincere thanks to my mother who has been my greatest support and especially my sister and brother who had motivate me to work hard in my work. I couldn't reach to this completion without the blessing of my mother and her abundant love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In cloud computing and virtualization study area, many new study subjects are available for Live migration and virtualization. There is no new and devoted solution to achieve high availability with live migration using Docker container. Docker container offers a lightweight technology in the virtual cloud network to deploy cloud applications. In IT industry and cloud networking platform Network operators have a less knowledge and awareness about Docker container technology and its benefits. Cloud computing is growing technology in the era of computing, which makes the organization to migrate to cloud platform. Buyya et al. (2009) state that, the word cloud computing denotes to the services which reaches or provided to the user based on their requirement and can access the services from anywhere and using any computing devices expressed cloud computing is provisioned by the interdependency of distribute computing and parallel computing in the virtualized environment. Cloud computing environment consists of three fundamental services they are Software as services(SaaS), Platform as a Services (PaaS) and Infrastructure as services(Iaas). In Infrastructure as Services, all the resources such storage, network topology, server space that make up the success of cloud computing are provided in either physical or virtualized hardware. In a cloud computing environment, virtualization being the backbone behind the success of cloud computing many of the organisation had been taking care of this concept. Building virtualization in cloud infrastructure will bring enormous advantages to the cloud service provider to lower down the investment and to provide sufficient resources to multi-tenant. In our research, we will be focussing on performance of cloud infrastructure deployment by taking the advantages of Docker container technology and solving the networking issue face by Docker in multiple cloud platform .

Docker containers are advanced in features when compared to Virtual machines and Linux containers. Even though Docker is a advanced version of Linux container they

facilitates the user in terms of easy portability, reduce libraries, and easy migration. Many testing and developing environment uses docker container because it is easy to spawn a virtual machine in few seconds. It favours the enterprise by deploying single standalone application for a container, whereas it leaves a challenge to the cloud architect to use docker for build a complete infrastructure. The main challenge is random IP allocation to containers. The docker allocates the IP from set of available IP in the pool if we shut-down container A and create a new container called container B, the new container gets the IP address of the container A(i.e. the container A in not active). To overcome networking issue many third party network plugins are available to communicate to the outer world. But these plugins have their own configuration, such as own crypto, minimum hardware/software requirements and even expanding the infrastructure by live migration is not possible. Today there is no devoted and isolated Docker container solution for live migration. For Docker container platform, it is essential to find out new techniques to address goal of high availability and networking issues in Docker container with limited resources Docker container should resolve the network challenges and offer high availability in multiple-tenant environment. **"In our research we propose a novel solution to address the above issue and build an Docker container infrastructure by creating private DNS container and connecting all the containers to the DNS".**

We introduce new approach called DNS communication for the containers, we will divide our research into four different sections. The first section 2 will include the related work of our research which describes the basic concept of Docker container and Linux container, it will highlight on the idea of cloud infrastructure management, finally the basic concept of high-availability and Live migration.

Furthermost, the second section is the Design and specification 3 chapter. In the first part of this chapter, we will discuss about our proposed architecture on how the Docker container infrastructure is built by creating a DNS server container and how the containers are connecting on all the containers to the private DNS server. Then in the second part of this section we will describe the specification or the requirement to achieve our goal.

In implementation 4 chapter, the section is divided into two part. The first part of this section deals with the infrastructure deployment using weave networking plug-in and its issues, the second part will describe the implementation of our prototype by building private DNS server and configure in master and slave pattern, finally the MongoDb is deployed and is connecting to the DNS server.

Finally, In evaluation 5 section we will perform various test on our proposed cloud

infrastructure which is deployed in multi-cloud platform. It is analyse and compare the performance with the existing infrastructure. In our research, we will deploy the MongoDB clustered architecture with our propose infrastructure. The comparison is done in order to show the better cloud deployment infrastructure among multiple cloud platform for the production deployment. Additionally, we will also generate the result, some of the observations that is recorded during the analysis.

Contribution

As discussed, cloud infrastructure management solution can achieve high availability and solve networking issues using Docker container. Our main aim is to achieve high availability across the multiple cloud network/vendor. Our significant contribution in this research is that :

1. This solution will successfully deploy infrastructure across multiple cloud vendors. We will deploy MongoDb clustered architecture with our proposed infrastructure. We will deploy router and config component in one server,the shards component part we will deploy in another different server. We will make the connection between the containers with the help of our proposed private DNS server. We will test the performance of the infrastructure and compare to the existing infrastructure.

2. Our solution will offer high availability and can easily expand infrastructure without shutting down the servers.

3. Resource utilisation is less in term of CPU and memory utilisation.

Chapter 2

Related Work

This chapter gives the related work and literature review regarding Docker Container and virtualization. In section 2.1 we will discuss about, cloud computing paradigm and virtualization technology. We explain cloud services and their related work. This section includes different virtualization techniques to work with Docker container. In section 2.2, we describe in detail about Linux Container (LXC) and Docker container. This section give previous and related work of Docker container and LXC. In section 2.3, we focused on cloud infrastructure management and previous management techniques. In section 2.4, we point out live migration work and how it affects on high availability of virtual machines.

2.1 Cloud Computing and Virtualization

Cloud Computing (CC) is a concept of computing service as a utility. Armbrust et al. (2010) state that, computing utility has the ability to improve performance of IT industry and provides attractive software services. In a utility computing study Armbrust et al. (2010) discussed about, Internet services and how multiple servers are used to manage scalability and costing in the industry. Liu et al. (2011) designed cloud computing service model for National Institute of Standards and Technology (NIST). According to NIST, cloud computing definition includes cloud services: cloud software as a service (SaaS), cloud platform as a service (PaaS), and cloud infrastructure as a service (IaaS). CC also categorized four deployment models: private, community, public, and hybrid cloud.

Dong et al. (2014) state that, virtualization is a basic building block of cloud computing and data centers. Virtualization is able to run multiple guest Operating Systems (OSs)

on the top of single physical machine. Dong et al. (2014) paravirtualization technique includes software coding solution for Memory Management Unit (MMU) and I/O. Dong et al. (2014) introduced hybrid virtualization solution with hardware virtualization in it and results on performance improvement. Dong et al. (2014) argues that, to paravirtualize MMU it's important to implement guest page table using the guest memory address and the physical address of the host. Dong et al. (2014) further state that, Direct Page Table (DPT) entry is important to monitor approve security for the hypervisor.

In study of network virtualization with high performance Dong et al. (2012) focus on, single root I/O virtualization (SR-IOV). In this technique I/O device can be shared by various Virtual Machines (VMs) with balancing high performance. He propose a generic framework for virtualization to share SR-IOV. This design can be implemented on multiple hypervisors. This experiment includes SR-IOV test to check their performance. Test result achieves 9.48Gbps line rate and scalability over 60 VMs. Dong et al. (2012) compare this result with network driver in paravirtualize environment. Dong et al. (2012) argues that, this study offer minimum CPU utilization than paravirtualize network driver.

2.2 Linux Container (LXC) and Docker Container

In introduction of Docker Boettiger (2015) state that, Docker is more integrated with Operating System level virtualization, module re-usability and Development and Operational (DevOps). The DevOps is a agile methodology to establish communication between development team and operational team to offer good quality product (**?**). The reproducible research is based on a number of scientific processes like, collecting data and processing it, analysing and visualizing data. The algorithms are used to do computation process and conclude results. These combinations of Docker are significant to deploy cloud applications. This study is used, to solve dependency hell complication using Docker. This issue is needed to be considered during software installation. Docker can easily solve this complication by implementing binary image with an integrated software policy with configuring it(Boettiger, 2015).

Liu and Zhao (2014) state that, virtualization is the fundamental structure of cloud computing and Docker technique is use for virtualization in cloud computing. Liu and Zhao (2014) focus Docker virtualization model in their study. Docker is an open source framework for developers to deploy applications on distributed cloud systems. Docker

is developed in Go language with important Linux kernel characteristics. Docker includes namespaces and to isolate layers of containers. Process ID, Networking namespace, communication ID, mounts ID and Unix timesharing system ID are some of the namespaces used by Docker. The libcontainer is a default format for containers, which also integrated with LCX. Liu and Zhao (2014) stress that, dotCloud is one well known service in PaaS and supports multiple cloud computing applications. These applications can deploy using open source platform of Docker. Docker is allowing us to use multiple resources with Docker engine and avoid to developing unnecessary apps to under PaaS. Docker can design images on requirement using important instructions present in Dockerfile. Dockerfile includes all instructions need to execute Docker image. The Docker engine runs the container on Host OS and Host will then push the container's image on the network. So the other nodes on the network can easily pull Docker image and create multiple containers.

To develop cloud platform architecture Liu and Zhao (2014) suggests, private PaaS framework using the Redis tool. Redis is also an open source project and used as a data structure server. According to the requirement of data, the system allocates various computing functions to various platforms. Dusia et al. (2015) indicate the issue of sharing network bandwidth and offering Quality of Service (QoS) to users. The single bandwidth application is responsible to degrade the performance on a network where Docker framework is used. Dusia et al. (2015) point out that, Docker containers significantly increase the user experience with sharing network bandwidth and assure about Quality of Service (QoS). To provide high quality applications for Dockers some services are needed to fulfill network requirements. Using QoS Docker can ensure about network bandwidth with matching priorities. QoS offers two benefits: User applications who need high bandwidth, Docker can assign them high bandwidth. The second benefit is, existing resources of the network can use to decrease the operational costs.

Liu and Zhao (2014) specify framework benefit of Docker that it is able to use a special type of technology known as the control group. Liu and Zhao (2014) further describe that, containers are famous to offer good performance in multi-tenant network architecture and cloud applications easily access selected resources. Rey et al. (2015) argue that, evaluation and performance testing is very difficult task in distributed cloud computing system. Rey et al. (2015) focus on Docker container based virtualization technique which is based on Hadoop and is used to deploy applications such as MapReduce. MapReduce is basically used for parallel programming design to do computations on huge data sets. The main advantage of MapReduce is, it uses a high degree of parallelism in programming language to process Petabytes of information in a dedicated amount of time on huge clusters with multiple machines.

Rey et al. (2015) explain about PER-MARE tool, this is used to maintain fault tolerance capacity of network design. PER-MARE is a tool to transform the Hadoop database architecture into the case pervasive network. This study offer a fault tolerant software tool and give a solution over the flaws of Hadoop. This tool is able to stop job scheduling when network fails to schedule jobs. Rey et al. (2015) further state that, Docker-Hadoop significantly improves scalability of the network and achieve high speed. The Docker-Hadoop design framework allow rapid deployment of a data of a Hadoop cluster. The container oriented virtualization offer the researcher to manage network nodes and virtual images of the system. To increase the usability of the framework, user can add management interface to create new nodes by just clicking on it.

In a virtualization study Raho et al. (2015) offers, layer of hardware abstraction using an implementation of virtual machines (VMs). This research is mention about containers, that are well known for virtualization these days and use as replacement technique for virtualization in resource sharing. This research is on open source hypervisor tools and compare them as a solution for the ARM architecture. These tools include KVM, Xen hypervisors and Docker container.

Raho et al. (2015) conclude that, KVM hypervisor is certainly better than Xen hypervisor in case of ARM architecture integration. For testing Raho et al. (2015) used Hackbench benchmarking tool for evaluation and testing purpose and achieved negligible overhead for Docker.

Bernstein (2014) also intimate that, Docker containers are made up of base images using apt-get install command. apt-get install is responsible to build new layers of Docker image on old image. Dockerfiles are used to execute this newly created image of Docker. Ismail et al. (2015) focus on the issues of cloud computing, such as congestion control, performance bottleneck and high bandwidth latency. The study is done on experiment and test Docker container technique and implement for cloud computing application framework. Docker is divides into two use cases, namely: continuous integration and other is continuous deployment. These cases are important elements of computing framework. According to the observation, Docker is designed to create a single application loosely coupled with one container. Docker gives better results with low system footprint. Ismail et al. (2015) conclude in his experiment that, application deployment with Docker container is fast. Docker does not give memory footprint and minimize performance bottleneck of computing system. They also point out that, Docker implementation is based on two cases such as continuous integration and second is continuous deployment.

In a technical study of Docker Ismail et al. (2015) conclude that, Docker have some

network issues need to be addressed. But there are many research areas for Docker study and space to achieve high performance. Docker offers good quality deployment, better performance on virtual machines based computing platform. Docker deployment technique is more suitable than virtualization technique. Bacis et al. (2015) confer that, Docker is used to create and retrieve multiple virtual images of the source machines. Security of Docker is a significant element that needs to be studied and evaluate. The Docker is also uses Linux container (LXC) characteristics such as cgroup, access control and LXC namespaces. These features are specially used for security purpose. According to Bacis et al. (2015), Docker framework need be more secure because it manages the virtual environment to acquire virtual images from source machines.

Bacis et al. (2015) introduce a framework to offer flexibility and secure design to Docker container with Dockerfile file format quality of docker. This framework create and run virtual image of Docker using SELinux and give secure environment to Docker. Dockerfile is used to create a Docker image, which is then configure container. Docker image can be pulled from other images on the network repositories. Unlike traditional virtualization, Docker does not use the new OS to create a new image of the system. Linux services are generally used to separate the containers. Bacis et al. (2015) further state that, SELinux is used to provide sound based security and also provide policy modules to give service improvement. The Linux container is a operating system level virtualization technology. Bacis et al. (2015) further assert that, image maintainer should take more benefits using SELinux and extend policies using DockerPolicyModule.

In study of LXC Bernstein (2014) Docker is an open source tool and used in the virtual framework to deploy Linux applications. All Linux applications in virtual network are deployed in the container itself. Containers are popular now a days, because they can easily share applications with the OS and deploy them in a small file size. This application deployment is smaller than deployment with virtualization deployment over the Hypervisors. Bernstein (2014) point out that, Docker container helps to expand Quality and functionality of Linux container (LXC) using core kernel layer and Application Programming Interface (API). Each Dockerfile is nothing but script of instructions and list of actions to build new image of Docker. Dua et al. (2014) also study about, virtualization and Containers and differentiate Virtual Machines (VMs) and containers on the basis on Guest OS, Security, machine performance, execution time and storage capacity. Dua et al. (2014) mention that, container's performance is better than VM's performance and it takes less execution time.

According to Bernstein (2014), many cloud computing frameworks in IT industry uses Virtual Machine Monitor (VMM) to crate virtual environment and deploy cloud applications. Dockers can work with virtual machines also. Applications those, who run

only on virtual machine with Hypervisor, can also place and run in Docker container. To et al. (2015) state that, Linux Container is based on server virtualization and used to divide host Operating System (OS) to make virtual instances. As VMs are important in the cloud computing framework to allocate multiple resources to users. Network bridge in Linux platform offers virtual network switch to connect multiple nodes together.Felter et al. (2015) argues that, Linux containers gives a better performance measurement with VMs and able to deploy cloud applications with high performance. In experimental set up they compare, KVM hypervisor and Docker container. Felter et al. (2015) conclude in result that, container performance achieved good result than VMs. The main goal is to separate and measure the overhead of all the VMs and Dockers. The Linux platform can manage virtual machines as well as containers to compare both platforms.

Felter et al. (2015) confer that, containers are behaving responsible like VMs and Docker increases performance measurement than KVM. In network virtualization study Casoni et al. (2013) assert that, state of art is one effective solution for network virtualization and used to simulate it. Casoni et al. (2013) test VALE virtual bus, on a small network to interconnect network machines and implement packet transmission across nodes. Linux container plays a significant role to establish packet transmission over network with the help of the VALE virtual bus. The study compared this VALE solution with Virtual Bridge and Virtual Ethernet interfaces that is included in Linux category. Casoni et al. (2013) conclude that, VALE bus solution is effective for high traffic networks and can achieve high scalability with network virtualization.

To et al. (2015) developed an emulation tool called Dockemu to create a strong network model that gives significant way to emulate the network. To et al. (2015) mention that, Dockemu is build up with features like Linux Containers and Linux Bridge. Dockemu is used to emulate layer 1 and layer 2 of OSI (Open Systems Interconnect) model. The Dockemu tool can be used to emulate wired network as well as a wireless network. Dockemu is a special tool for network emulation. To et al. (2015) argues that, in coming years Dockemu tool can use with cloud to create elastic cloud computing framework.

According to (Liu et al., 2011), Platform-as-a-Service (Pass) service vendors of cloud computing have many issues regarding service provisioning. LXC can help to resolve such issues for PaaS vendors. Dua et al. (2014) explain of virtualization and container concepts and about some PaaS use cases. The container based PaaS designs are given better performance and use full for well established vendors. A container has light weight OS and instructions for CPU to avoid instruction level emulation. Dua et al. (2014) also discuss bout different containers such as LXC, Docker, OpenVZ and Warden container and their features. With Platform as a Service developer can give a product to

deliver services quickly on IaaS. Docker containers are usually used to reduce overhead. A container is responsible to separate a particular process from the rest of the machine processes. Dua et al. (2014) conclude that, containers have significant features to improve performance and decrease start up time. There are many different ways in open source platform to implement container in the system. Docker is used to add one layer on the top of Linux layer, so it's easy for PaaS vendors to deploy applications. Containers allow a good research space for PaaS vendor using their use cases.

Mattetti et al. (2015) accomplish the tests to achieve high scalability for cloud applications using Linux Container. To achieve these elements in cloud infrastructure the important thing is that, security techniques should be strong and able to protect whole network framework, the reason behind that is application deployment is happening inside the container. Mattetti et al. (2015) introduce a LiCShield tool to provide a secure framework for container to manage workload.

2.3 Cloud Infrastructure Management

According to Marquezan et al. (2014) cloud infrastructure management is a difficult goal to achieve. Because of it's complexity and multiple layers of cloud framework, cloud infrastructure needs management methods. Marquezan et al. (2014) further mention that, for this complex structure when we apply management methods it's important to maintain the quality of cloud application. Marquezan et al. (2014) develop one a 3-D monitoring tool to analyse the cloud performance. The research conducts experiment on Openstack platform. The results showed 3-D monitoring tool achieve successful management methods for identification in cloud framework.

In study of cloud infrastructure management Yan et al. (2011) state that, IaaS is most popular cloud computing framework. But IaaS has some issues in the hybrid cloud environment. The study proposed a solution for this problem named as Monsoon. Monsoon is used to satisfy cloud requirements of enterprise users. Yan et al. (2011) argues that, Monsoon provides user interface and portal to manage cloud infrastructure. Yan et al. (2011) further state that, Monsoon includes key services like management tools, access control tools and analytic tools. The study also mention that, implementation engine in Monsoon gives user ability to define security, industry and government policies and other requirements.

In study of cloud key management Lei et al. (2010), discuss about cloud data protection using strong encryption key management. This proposed solution is called cloud key management (CKMI). The main function of CKMI to minimize the complexity

between clients and servers in cloud framework.The client uses cryptographic services and servers are based on strong key management. Lei et al. (2010) claim that, CKMI offers the ability to reduce risk and cost to cloud infrastructure. To manage cloud networking framework Dhungana et al. (2013) give a solution with User Managed Access(UMA)protocol. Dhungana et al. (2013) state that, this design offers authentication, identity management and authorization to cloud network. UMA is used to provide elasticity and security in the virtual cloud network.

2.4 High availability and Live Migration

In study of live migration using Docker containers Yu and Huan (2015) state that, this technique is lightweight and mainly use for VMs. Yu and Huan (2015) point out that, Docker container reduces the time require for live migration. Now a days most of the data centers are using lightweight virtualization technique and implement it on the Docker container to achieve high availability. Sun et al. (2013) provides a solution, called live snapshot technology for cluster based computing. According to Sun et al. (2013) live snapshot provides high availability to VMs. They proposed Selection Concurrency Strategy to reduce the downtime of VMs.

Liu et al. (2014) state that, to achieve high availability cloud infrastructure management plays an important role. The study provides a benchmarking, measurement tool to measure availability while deploying cloud stack. Jin et al. (2013) justify that, VM's live migration have some benefits like high availability and load balancing. Jin et al. (2013) introduced, Partners Assisted Storage Migration (PASM) technique. This technique is used to lively migrate shared storage and save 78.9 percent migration time with high availability. Patnaik et al. (2010) state that, if live migration of VM fails, then it's necessary to migrate it on another physical machine. Patnaik et al. (2010) provide a solution with IP communication deployment on virtual environment with high reliability and availability. Patnaik et al. (2010) claims that, this solution is useful for service providers in telecommunication industry.

Ahmad et al. (2013) discuss about live VM migration in secure cloud framework. They carried out security requirements for live VM migration. In another study of live migration Fakhfakh et al. (2009) point out that, telecommunication sector is also using virtualization technology from last 20 years. The OS instances can be migrated on different physical machines to avoid hardware failure and achieve high availability. Fakhfakh et al. (2009) further state that, in 3rd generation Internet (3G) world live migration is helping to minimize overhead issues in virtual network and telecommunication industry. To analyse the performance of application Anala et al. (2013) introduce

one solution using virtual machine. Anala et al. (2013) describe that, live migration is act of migrating Virtual machines from one physical device to another without degrading performance and interfering service. Anala et al. (2013) mention that, network administrator is responsible to migrate particular VM and this decision is depends on availability rate, downtime, migration time and page dirty rate. As we have discussed in this chapter, the previous work of docker and Linux container.We have also see the argument made by many of the researcher with regards to the networking issues of the Docker. Many researcher and the third party they study about the various solutions to the problem. However,there is no specific solution that could solve the networking issues. In the next chapter of this study, we will be discussing in brief about the design part of our project.Where it also give the brief overview of the architecture of the private DNS connectivity for Docker container in multiple cloud environment.The main area we are focussing is the improvement of the connectivity of Docker container. Moreover,there is no particular solution ,the question that arise here is that How can the Docker container connectivity be improved ?.In our research,we will give the solution to the problem by implementing private DNS in Docker container that will be the connectivity bridge between the Docker Containers.

Chapter 3

Design and Specification

In this chapter, we will explain in detail the proposed infrastructure design of our research. We will also explain the hardware and software requirement of each tool and technology we used in our project. Further, the last section of this chapter, we will define, in brief all the tools we will be using for testing the result of various performances which we will implement in our project.

3.1 Design

As we have described in the literature review, that at presents docker container is said to be the growing technology in the IT industries. Therefore cloud infrastructure being the backbone need to be taking into account. Cloud infrastructure deployment of containers will help achieving high availability and low resource utilisation, since we can build the infrastructure in low resources among multiple cloud platform. The question that will arise in this is how can cloud infrastructure be built in low resources to achieve high availability in different cloud platform. Our primary goal of this research is to test the performance of cloud infrastructure deployment, among single cloud and the among multiple cloud vendors by placing our private DNS running inside the container. They act as the connector between the three hosts as shown in the figure 3.1. The three cloud vendor, we have been using in our research is Digital Ocean, Open stack and AWS (Amazon Web Services) and we deploy the MongoDB component in each cloud vendor to check their cloud infrastructure deployment performance and test the result, however this will be described in the next chapter of the document. In order to generate the performance result of the cloud infrastructure we will be using YCSB benchmarking tool and New Relic to analyse its resource utilisation.

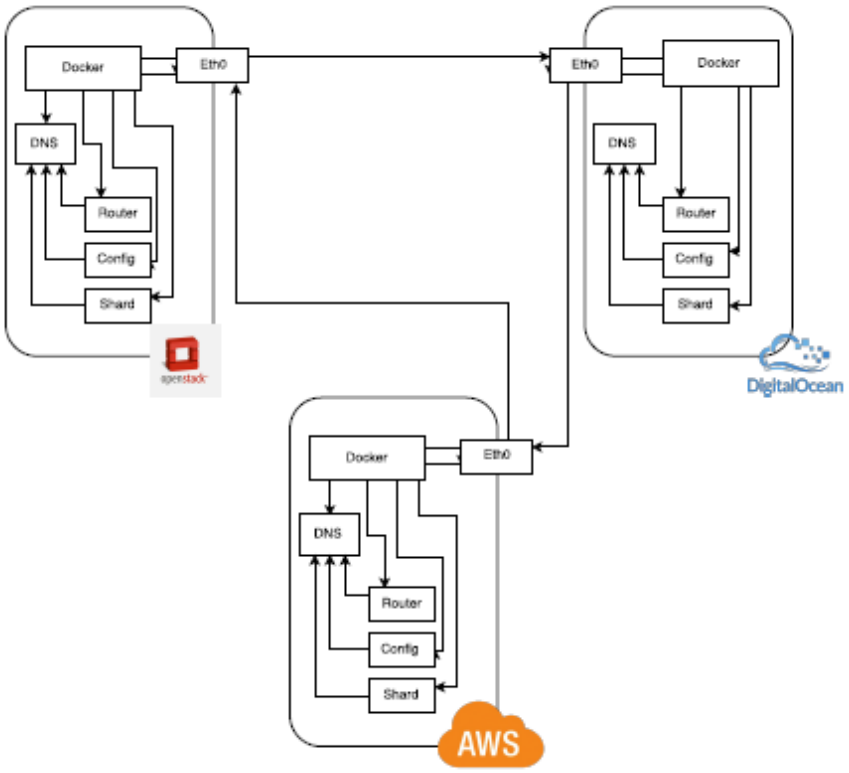

Figure 3.1: Proposed architecture.

3.2 Specification

In this chapter, we will demonstrate the tools requires for implementing the prototype of our project with regards to improvement of MongoDB performance by deployment of cloud infrastructure and achieving low resource utilisation.

3.3 Software Requirement

3.3.1 MongoDB Sharding Cluster:

MongoDB is an open source document oriented NoSQL database system which is written in the C++ programming language. The main vital function of MongoDB is that it manages collection of BSON documents.MongoDB development began in October 2007 by the organisation known as the 10(gen). MongoDB holds special features such as document oriented storage, JSON-Style documents consists of the dynamic schemas that avoid complexity, full index support, replication and high availability. MongoDB

also support auto-sharding, GridFs, store files of any size without any error to the stack.In the centre of MongoDB lies the concept of the document which is the primary unit of data for MongoDB, when the request is made from the client,the mongos or the router routes the request to the matching server and merges the result to send back to the client. In our research, we will be focussing on the deployment of MongoDB cluster and we will discuss in detail in the next section.

3.3.2 MongoDB clustered Architecture:

Mongo DB cluster provides horizontal scaling out of the database in very low cost, usage of commodity, hardware or cloud infrastructure by using the technique called sharding. Sharding evenly distribute the data across multiple physical partition called as shards. With the concept of sharding in MongoDB helps to address the number of hardware limitation of a single server, such as bottleneck in RAM or disk I/O Without adding any trouble to the applicationLiu et al. (2012). The figure 3.2show the architecture of the MongoDB sharding architecture.

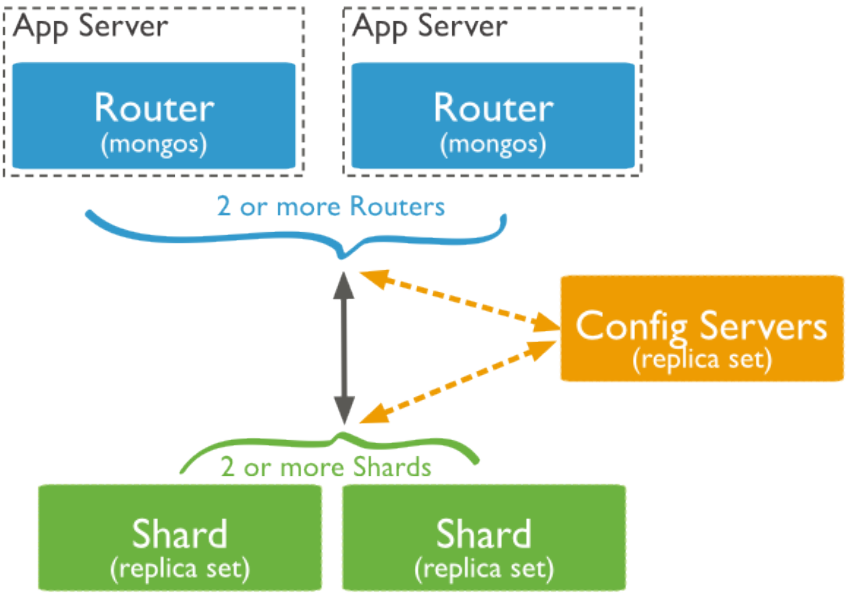

Figure 3.2: MongoDB Sharded Architecture.

The following section, described the different component of the MongoDB shard clustered architecture:

3.3.3 Shards:

Shards are said to be a single mongod that consists a subset of data for the shared cluster. Collection of these clustering shards holds the whole data set for the cluster. In general, the shard is replica set which generates redundancy and high availability for the data in every shard. Every shard will consists of one or multiple server and it uses mongod process for storing data.

3.3.4 Config Server:

Config server they mainly store the metadata for sharded cluster. MongoDB make use of config server to maintain the distributed lock. Furthermost, this component is used by the query router where they use the metadata of this component to target the operation to particular shard. Production sharded architecture cluster they mainly consist of three config servers, however, for testing purposes we can use one config server.

3.3.5 Router Server:

The other name of this component is the query router as it used to route the process and coordinate with the entire process taking place in the sharding cluster architecture. For instance, when the client query to the MongoDB server, router server will routes that particular request to the particular server and match with the needed result and sent back the request information to the client. Sharded cluster can have more than one router for balancing client query load.

3.4 DNS Server:

Domain name system(DNS)is one among the most critical part of internet in todays generation, especially in the IT industry, where it provides links between user and internet location by associating the host name to the IP addresses with the help of various protocols like TCP/IP and many more Satam et al. (2015). DNS technology has much more useful with the evolution of cloud computing, in which the cloud are in need of DNS servers to be more dynamic so that it can adapt to any new services which comes in the market based on the user demands Jalalzai et al. (2015). As cloud computing technology becomes overwhelming DNS protocol plays the highly important role for the better performance of every services provided by this technology. All this

service could be the proper functioning of the protocol to be used for DNS links in order to achieve proper transfer of data from one server to another without any lost, quick response of user queries and many desired services which is based on the users demand. In our research, the DNS server will play a very vital role in which it will be used as the main source component for networking the container together among different cloud vendors or even in different networks,

3.4.1 DNS Zone

DNS is a technology that allocated the naming system by using a set of hierarchically connected DNS server at the time while maintaining a client server model. It is mainly used to transform the human readable domain names to IP address to be readable by the system. The DNS functions by passing the query hierarchically over a tree like the domain space network via protocol, it then travel to the destination network till it reaches the server which have the authorities for the requested domain. Then the server that acts as the destination will send the reply down the tree to the source which make the request with the DNS reply. The tree networking structure in DNS is further subdivided into smaller network known as the zone beginning at the root zone. A DNS zone is said to be well-defined part, critical portion of the domain name space in the Domain name system, it is said to be the root of the DNS which is administrative responsibility is minister only to single manager.

Every individual zone has at least one authority name server that provides permission response for that particular zone. Each zone may have more than one domain, therefore some of the servers that have the permission may sometimes delegate its task of responding the request or the query of its zone to some other DNS server that have permission to the sub zone. Generally, the network host consists of the list of root authority's server, the server is then passes the query that travel all the way up the tree till it reaches the appropriate DNS server which will provide the response to the request and send back to the local DNS server and in turn sent to the reply to the requesting machine resolver Chandramouli and Rose (2005). DNS reply it actually depends on the Time to live(TTL) and the types of DNS server involved. In our research the DNS server we will use is called as BIND (Berkeley Internet Name Domain). The main reason of using this DNS server is that it can be easily modified in case if we need to add any functionality in the future, this can be done easily as it is a complete open sources. BIND is also set to be one of the most flexible, full-featured DNS system. We will describe Bind in the next section of this chapter.

3.4.2 BIND DNS:

BIND stands for Berkeley Internet Name is among the most known software for converting domain name into IP address which is usually found in Linux server Liu and Albitz (2006). In this chapter, we will explain in detail on the basic concept of DNS BIND and briefly explain on various files associated with it for setting up our own Linux DNS BIND server. This software help user to do configuration in a very simple way, as mentioned earlier BIND server software,it is open sources, therefore on setting up BIND we need to download from its URL and install on our Linux server that it will provide the ability to become a private DNS server. It is stated by Deb et al. (2008) maximum of the DNS server is based on the BIND server because is said to be the best and reliable DNS server. While downloading this software, it approximately takes 4.8 bytes, and it needed to be compiled which is a very straight forward process. However, the steps for compiling the BIND DNS depend on the version of Linux we used and its distribution. The creation of the DNS server will be described in the implementation section of our research.

3.5 YCSB (Yahoo cloud serving benchmark)

YCSB is a framework for benchmarking systems. To work with YCSB we can either download the latest version from the source or we can clone the git repository. YCSB is used to compare its performance with other cloud platform databases. This framework consists of a package of a standard workload unit with varying parameters such as read-heavy workloads, scan workloads, write heavy workloads etc. The framework also consists of workload generators based on the parameters and limits provided. This is considered as the best part of YCSB. Because by using the workload generator we can generate and design any type of data needed for the analysis. This unique feature of adaptiveness in creating new data frames to benchmark cloud data serving systems is the key success for the usage of YCSB. This framework is available in open source so that many developers can use it to evaluate their systems a extend the tool and can also contribute new workload packages which would be helpful in developing new models. The important objective of YCSB is an extensibility so that is can be implemented in different new cloud database systems and benchmark it.

Architecture: YCSB is a Java program which is used to generate the data and operations which create a workload. This workload is then later testing on different cloud database platforms based on its performance. The architecture of YCSB client

is given 3.3 .

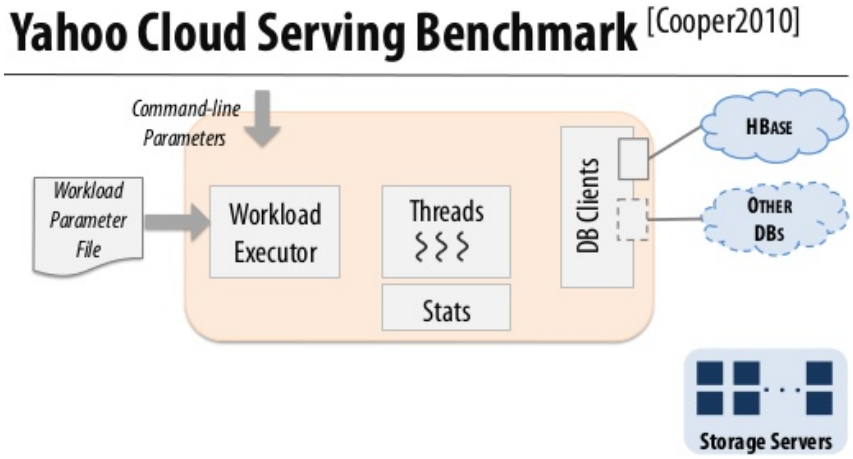

Figure 3.3: YCSB Architecture.

Initially the workload executor drives the multiple client threads on the database interface layer for both loading the database and executing the workload. The client threads, adjust the rate at which they generate the request to the database interface layer in order to control the offered load against the database. These threads measures the parameters such as latency, throughput, etc., of the operations to the statistical module. Finally at the end the statistical module will aggregate the average and present the result on a time series basis. There are two different kinds of properties available in YCSB. One is the workload property and the other one is the runtime property. Property which defines the workload independent of the given database are known as the workload property. Read and write proportions of the database falls under workload property. Properties specific to a given experiment is known as runtime properties. For example, Cassandra database interface layer to be used for a specific experiment falls under runtime properties of YCSB.The mostly workload property will remain static and runtime property might vary depends on each experiment requirements Cooper et al Cooper et al. (2010).

3.6 NEW RELIC

New relic is said to be a real time, on-demand application performance monitoring (APM) tools which are being used by many organisation. It is especially a tool used by Line of Business(LOB) professional, digital marketing experts or the owner of various e-commerce to monitor and understanding the Application Performance management

strategy. With the help of this tool the organisation can have visibility to the web user experience which is one of the basic roots of the success of the organisation. By using this tool the organization can monitor the CPU has been utilise, the memory usage, it can even help avoiding bottlenecks which lower down the performance of the application or any resources that it is monitoring. New Relic is the only SAAS web monitoring tool solution that allows user to check the performance from the number of web user experience, server performance, and its down to the line of the application code in the single result. This monitoring tools, it has many of its advantages as it easy to use, the result of monitoring the resources is generating very fast, and it is very quickly pays itself for many times over. Adding to this, New Relic models provide the organisation and its IT and web development the best function which is flexible for use, and it can be control to access the data from any browser, at their own needed time with no down time and help the organisation to eliminate the cost of infrastructure, support, in a traditional on-premises. In our research, we use New Relic in monitoring the CPU and the memory usage of the MongDB cluster, we will also monitor the performance of the server where we can analyse the number of Docker container being used. With the help of this tool we will analyse the amount of space being used in CPU and memory, when the processor workload is being loaded and run in the server. So the tool will give all the results when on the amount of spaces being used and the number of container left, this tool result helps the organisation to move or migrate the data to another server or we can add another server as the load increases it help to avoid bottleneck and it will help to improve the performance of the cluster. New Relic tools had design in such a way that it makes the user very handy to use. On creating the account, we just need to go to Menu bar of new relic and we then select Application name which need to be monitored. In our research we select name of the server as it will be monitoring the server then the server monitoring dashboard will be generated automatically every analytic function will come as a single result. The New relic Dashboard can generate result of various function like response time, it even monitors the end -to-end transaction tracing, the code level visibility can be drilled down from the dashboard to check the performance impact of particular code segments and SQL statements. The other monitoring features we will be using in our project provided by New Relic is generating the result of bandwidth in different cloud platform and will analyse the scalability of the Docker containers as the workload is increasing. We will also be monitoring the number of Docker container of the workload are running and how busy the system is, such information will be useful to analyse if our clustered database have enough container deployed to support as the load is increased. Security is one among the most important features provided by New Relic in making the resources which we will be monitoring to be safe and secure and to make sure the performance is

fast in generating the result. With the help of this tool organisation can also analyse the database call response time and throughput and check and search all database or cache operation within the selected time window Burson-Marsteller and Starch (2016)

Chapter 4

Implementation

In this chapter we going to look at the deployment of multi-vendor cloud infrastructure using docker containers. This section is divided into two divisions; first section says about the existing infrastructure that has been deployed using weave networking plug-in and its issue in adopting. Secondly is our prototype is that been deployed by creating DNS server inside the docker.

Prerequisites.
To complete the setup we the required following:

1. Virtual Private Servers from Different cloud vendors. We choose each server from Amazon Web Services, Digital Ocean, and OpenStack.
2. Bind9 DNS Server.
3. Docker Containers.
4. Weave docker networking plug-in(For deplying existing Infrastructure).

4.1 Deploying Existing Infrastructure:

We going look at the detail implementation of the container technology in the host server which is different networks. The Docker runs on the host and the container are created on top of the docker bridge. These containers receives the IP form the docker bridge. Docker bridge has pre-defined IP subnet that is 17217.0.0/16. These subnets are for internal communication but if two host have docker containers and the communication is not possible. Since the IP are not static and not private for the

config servers. The figure 4.1 tells no communication between two containers that are deployed across different network/clouds.

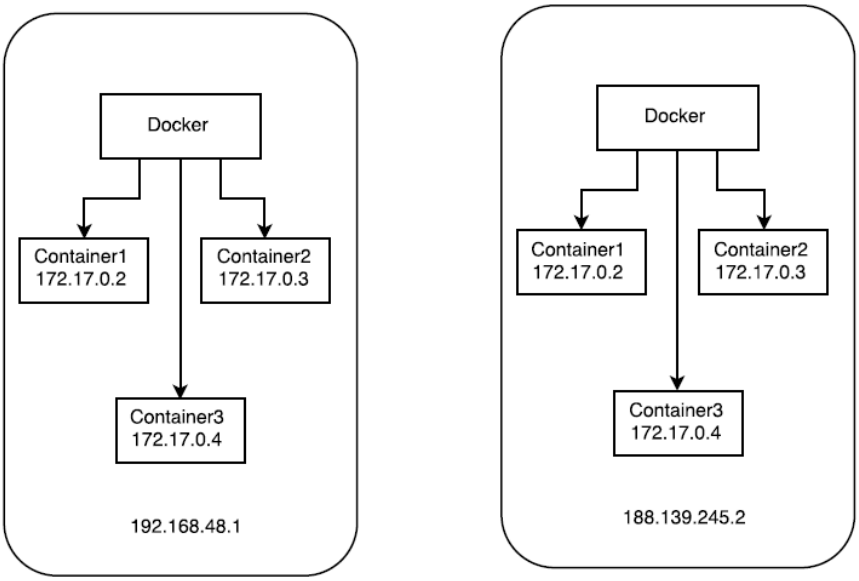

Figure 4.1: Servers With Docker Containers.

To avoid this network issue weave docker plug-in is been release to make the communication between the containers created at different networks. This network plugin in been install on the host and exectued as a another network bridges in the container such as weave and weave proxy. The weave plug-in provied the private IP to the containers and This plugin can be deployed for single server or multiple system. The below figure shows the architecture of the Weave netowrking. The disadvantage of the weave plug-in is that the plug-in will running on both host server and docker which more resourse will be utilized. Second the all the servers has to be configuired before the infrastructue is deployed, means that new server can't be added to the existing infrastructure.

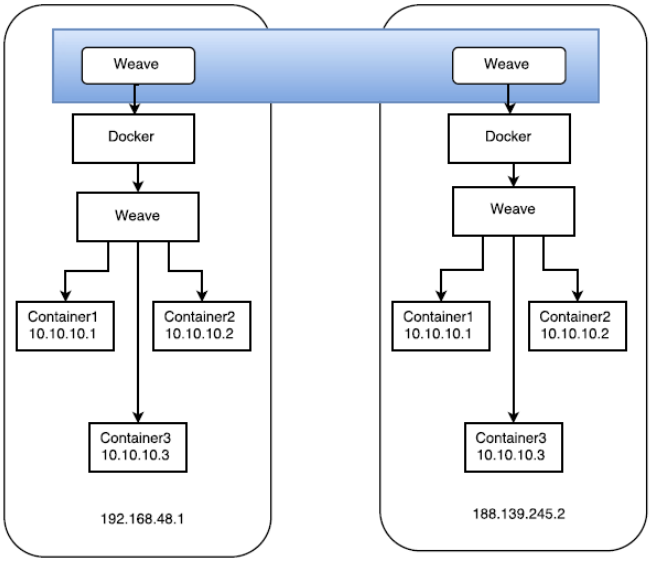

Figure 4.2: Weave Networking.

From figure4.2 we could see that weave create a layer 3(cloud overlay network) tunnel to establish the communication. If we have to add the new server to the existing one, the weave through an error that weave router has to stopped to establish the connection for the new server. Live migration of the process is not possible from existing infrastructure to new infrastructure.

4.2 Building an infrastructure using DNS.

In our prototype we are not using public DNS server such as Google DNS or Amazon DNS due to security concerns. Therefore we build our private DNS in a docker container.Before the setup we just bridge the servers eth0 to docker bridge to enable the communication. The Setup is categorised into three divisions:

1. Creating DNS containers in each server.
2. Creating the Containers and adding to DNS Server as nodes.
3. Deploying the Infrastructure.

At the first step we create a Docker container with Ubuntu 14.04 operating system and install Bind9 DNS server in it. Once the Bind9 package is installed, we have to go through three configuration scenarios such as cache server, primary or master server and secondary or slave server.

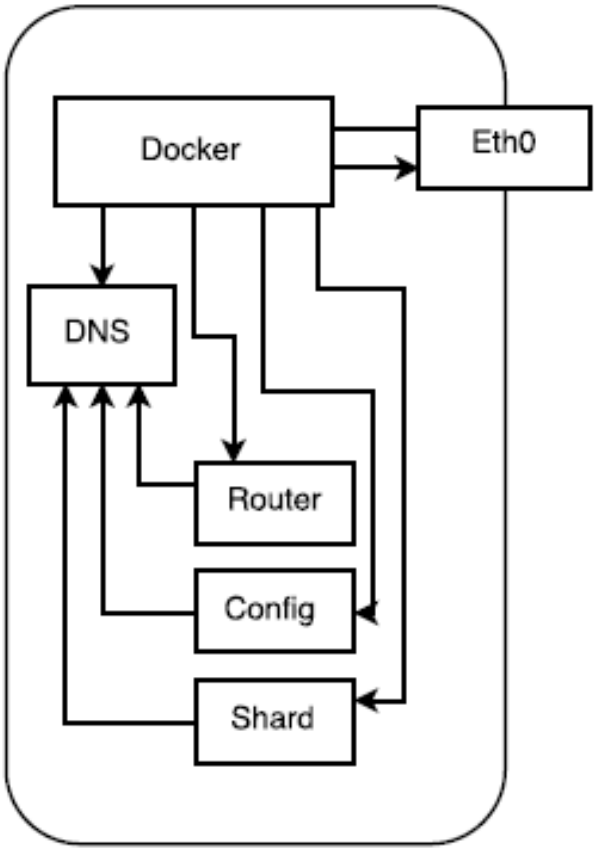

Figure 4.3: Configuring Primary DNS server.

4.2.1 Configuring DNS Server.

Primary DNS server has the group of records that are refered to the zones or registered domains. The below figure shows the creation of zone name called deploy.com in the named.conf in the primary DNS. After configuring the master dns we have specify the forwarding and the reverse zones.*Forwarding zone* configuration file is where we store the forwarding DNS lookup. This configuration can be defined in the new zone directory and save all the name lookups.In the *Reverse zone* Reverse zone is the place where we store the PTR records of DNS lookups.

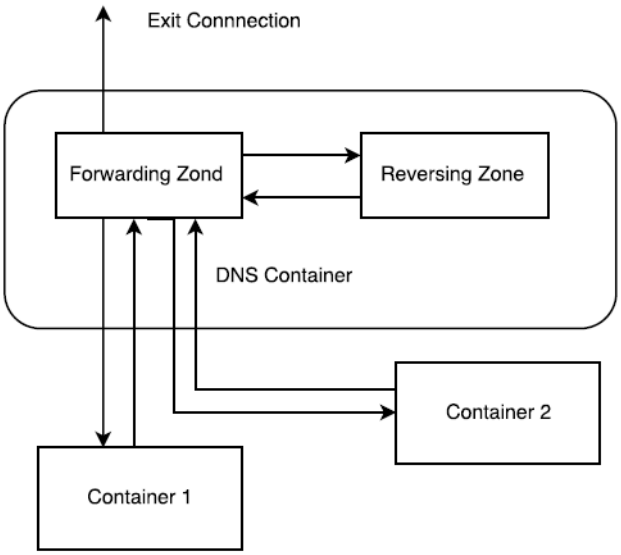

Figure 4.4: Configuring Primary DNS Server With Zones

4.3 Creating the containers and adding to the server.

After all the configuration of DNS is done, we can create the container and can be added to the DNS server zones.The Docker container are created using Docker file which is created with the set of the commands that will install the dependencies required and mongoDB application with the base image of Ubuntu 14.04. This script will make our installation easy and quick rather than executing the each and every command. The Docker file with set of command and containers created are provided below:

```
1  ####
2  ## Run this code on current directory of Dockerfile
3  ## docker build -t imagename .
4  ###
5  ## Enable container to run even after exit
6  ## docker run --restart=always -it -P imagename --name tagname
7  ###
8
9  # Step 1 - Base image
10 FROM ubuntu:14.04
11
12 # Step 2 - Replace shell with bash so we can source files
13 RUN rm /bin/sh && ln -s /bin/bash /bin/sh
14
15 # Step 3 - Add Author
16 MAINTAINER Karalyne Khongbri
17 # Step 4 - Mongodb installation
```

```
18 ##################
19 ##################
20 #mongodb key for version 3.0
21 RUN apt-key adv --keyserver hkp://keyserver.ubuntu.com --recv EA312927
22 ###authenticate the key
23 RUN echo "deb http://repo.mongodb.org/apt/debian wheezy/mongodb-org/3.2 main" | sudo ↩
       tee /etc/apt/sources.list.d/mongodb-org-3.2.list
24 ##update the linux
25 RUN apt-get update
26 RUN apt-get install -y wget
27 ###install mongodb
28 RUN apt-get install -y mongodb-org
29 ###make directories
30 RUN mkdir /data/
31 RUN mkdir /data/db/
32 RUN mkdir /data/db/config
33 RUN mkdir /data/db/shard1
34 RUN mkdir /data/db/shard2
35 RUN mkdir /data/db/shard3
36 RUN chown -R `id -u` /data/db/config
37 RUN chown -R `id -u` /data/db/shard1
38 RUN chown -R `id -u` /data/db/shard2
39 RUN chown -R `id -u` /data/db/shard3
40
41 # Expose port of container
42 EXPOSE 20000 10000 10001 10002 10003
```

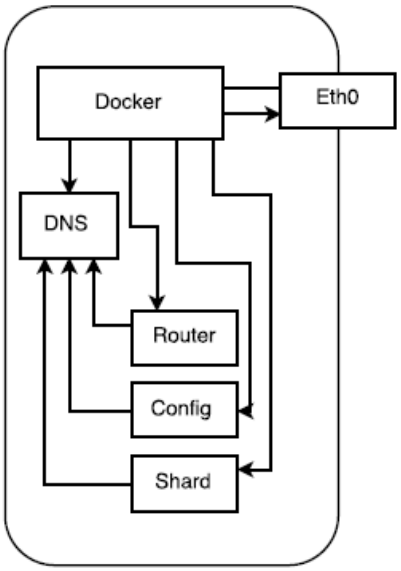

Figure 4.5: MongoDB Containers Connected To The DNS.

4.4 Deploying the Infrastructure:

We deployed a MongoDB cluster infrastructure with the domain name configuration across different cloud vendors.The complete deployed architecture is described in the figure 4.6. We have deployed high availability MongoDB server (i.e Replicated shard server). we have deployed primary DNS in Openstack Server and slaves in other two clouds such as Digital Ocean and Amazon Web Server respectively. This architecture provides high availability that can add new resources from other public or private clouds.

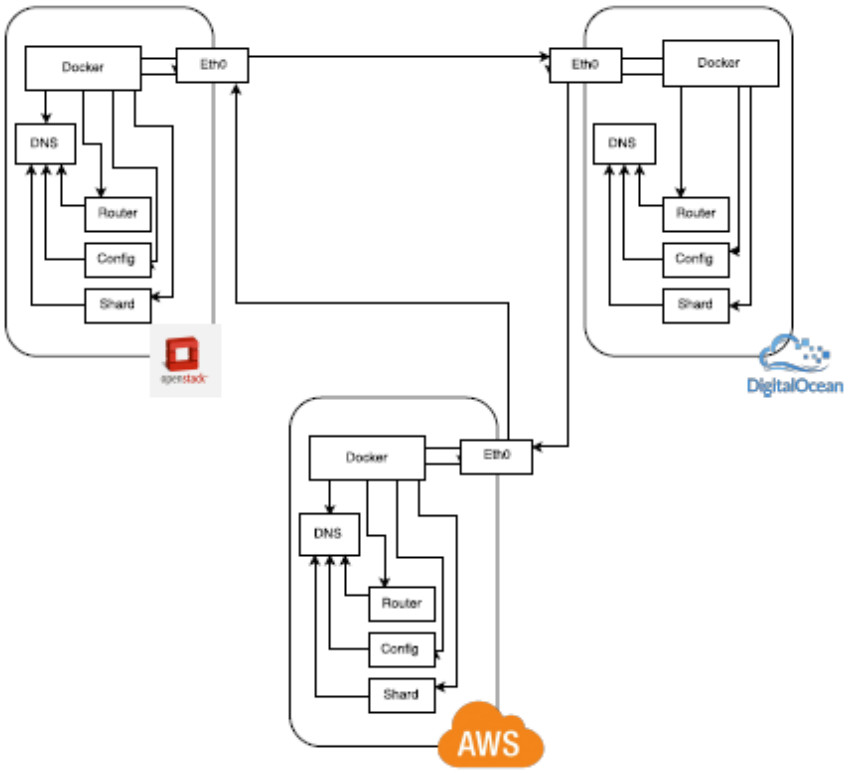

Figure 4.6: Infrastructure Across Multiple Cloud Vendors.

Chapter 5

Evaluation

As we had mentioned in the specification chapter,we make used of the YCSB tool In our project,we would perform the evaluation by testing our contribution in the following ways:

1. Will test the performance of cloud infrastructure deployment by deploying MongoDB Cluster according to our proposed architecture.We will generate the read latency time and the write latency time compare to the actual MongoDB clustered with our proposed model.We will test the performance of the infrastructure in two different ways:
   (a) Single sever.
   (b) Two server.
2. We will also test Our proposed infrastructure deployment architecture by deploying in multiple cloud vendor.We will deploy the architecture individually in each cloud and test the result performance of the infrastructure.Finally,we will generate the average result among the three cloud and make the comparison between them.The three clouds platform we will be using are:
   (a) AWS Cloud Platform.
   (b) OpenStack Cloud Platform.
   (c) Digital Ocean Cloud Platform.
3. We will analyse the resources utilisation of weave plug-in and DNS server when the cluster is deploy using both of the networking server.We will evaluate the resource utilisation into two different section:

(a) CPU utilisation.

(b) Memory utilisation.

4. We will also test the bandwidth availability between the Weave and the DNS server using the TCP protocol,the tested result can be measure using the QPREF tool which is the network monitoring tool.

5.1 Deployment Performance Testing in Single Vendor:

In this section we will demonstrate the testing results of the Cloud infrastructure deployment by deploying MongoDB clustered architecture in two different architectures in single cloud vendor. We will divide this section into three different sections:

1. We will deploy MongoDB clustered architecture, i.e., Router, MongoDB server and shards server in the single server and check the performance using YCSB benchmarking tool.

2. We then deploy the MongoDB cluster architecture according to our proposed model where we deploy the router and config server in one server,then migrate the two shards in the other server then we load the YCSB workload and run the workload and check the performance of the infrastructure.

3. We will compare the performance of both the model which will generate read, update and throughput latency in microseconds (μs).

5.1.1 Single sever:

As we had already mentioned, in this section we will be deploying the complete MongoDB clustered architecture in single server as shown in the figure 5.1. Follow by testing the performance of the model by generating the Read, Update and throughput Latency (s) of the architecture.

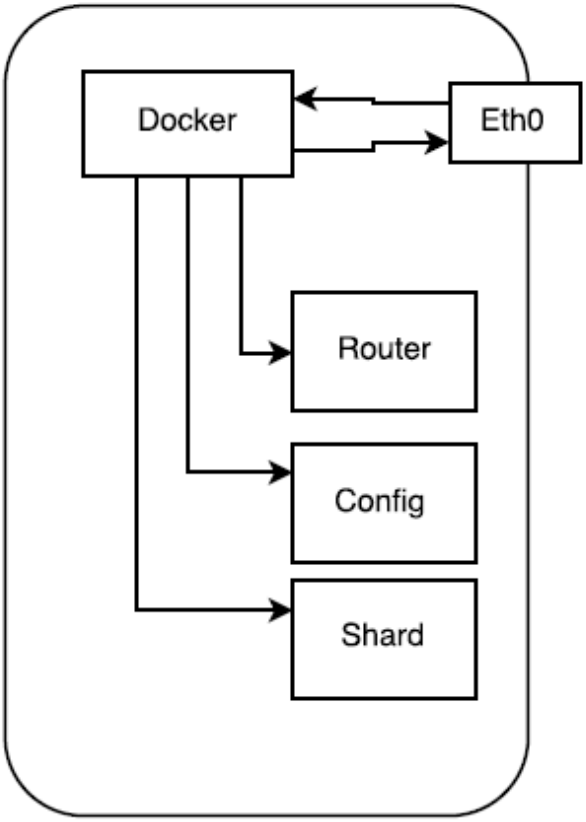

Figure 5.1: MongoDB In Single Server

The graph 5.1 generate the latency rate of the three operation as describe below:

1. **Read operation:**Read operation is the process in which the workload or the set of data is taking out of the database and the latency is checked and calculated the time it take for the data to be pulled out of that database.

2. **Write operation:** The write operation also known as the update operation is the process where the data or the workload is written into the database, likewise the write latency is checked by the time taken by the workload or write to the database.

3. **Throughput:** Throughput is defined as the measured of how many units of data a system can process at the given amount of time. In our research,the through put is define as the speed with which the YCSB workload is completed. The throughput in our research is measured and generate in microseconds (μs).

The table 5.1 depict the Test performance of cloud infrastructure of MongoDB clustered in single server, which we had performed by loading the YCSB workload and run the workload and perform the testing by running the load 5 times as we can see the Time taken by each infrastructure to operate the read, update and throughput in every test.

Table 5.1: Testing Performance Of Cloud Infrastructure Deployment In Single Server Model

Operation	Latency s
Read	2027
Update	2845
Throughput	429.4

The graph 5.2 depicts each testing performance of the MongoDB deployment infrastructure in a single server. On running the test 5 times, the read, write and throughput the result was consistence. In the next section of this chapter, we will see the deployment of cloud infrastructure of MongoDB in two different servers.

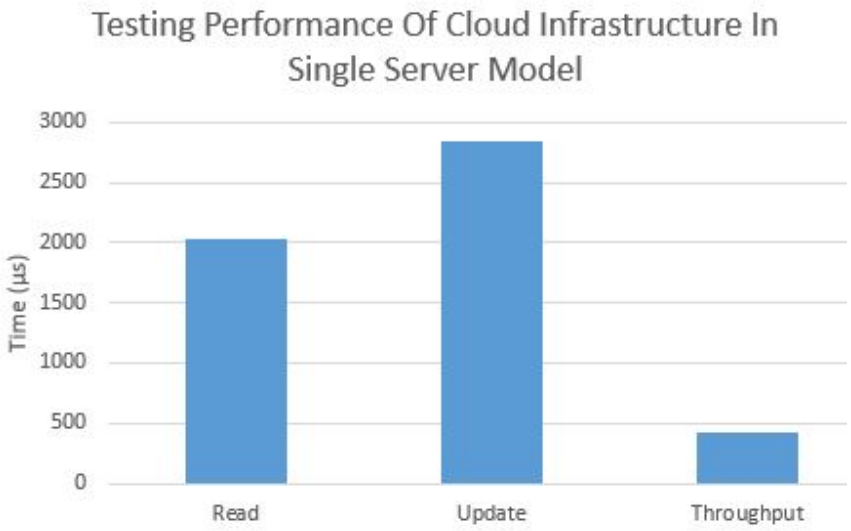

Figure 5.2: Test Performance Of Single Server Model

As we can see from the 5.2 the test result of the MongoDb Infrastructure which is deploy in the single server. The read, update and throughput operations performance is generated individually ,the throughput latency shows to be the highest of all the operations.

5.1.2 Two sever:

The figure 5.3 shows that we have shifted the storage/shard server of the MongoDB to another host so that the full resource can be utilized for the read and write operation. In this section we will describe the test performance of the cloud infrastructure of MongoDB as shown in the figure 5.2. In one server, we will deploy router and config server in single Host, we will move or migrate the shards in the other server as shown in the figure 5.3 below.

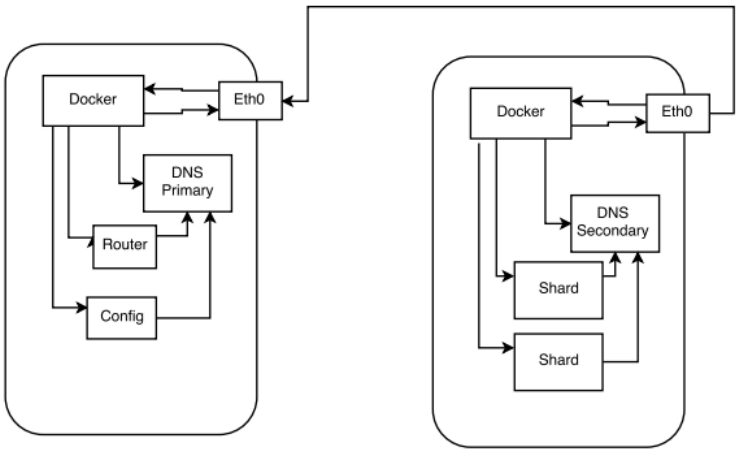

Figure 5.3: MongoDB Deployment In Two Server

As show in the table 5.2,we tested the read, update and throughput of the above architecture and we perform the test by running the load 5 times and generate the latency time taken by each test and each operation in microseconds(μs) per workload.

Table 5.2: Testing Performance Of Cloud Infrastructure Deployment In Two Server Model

operations	Latency s
Read	529.8
Update	772.2
Throughput	1115.2

The 5.4 graph shows the testing performance of cloud infrastructure deployment of MongoDB in two different Hosts. However, the test results which has been performing for 5 times does not have much difference with each other but remain consistence. The comparison of the performance of infrastructure in a single server and in two servers is given in the next section of this chapter.

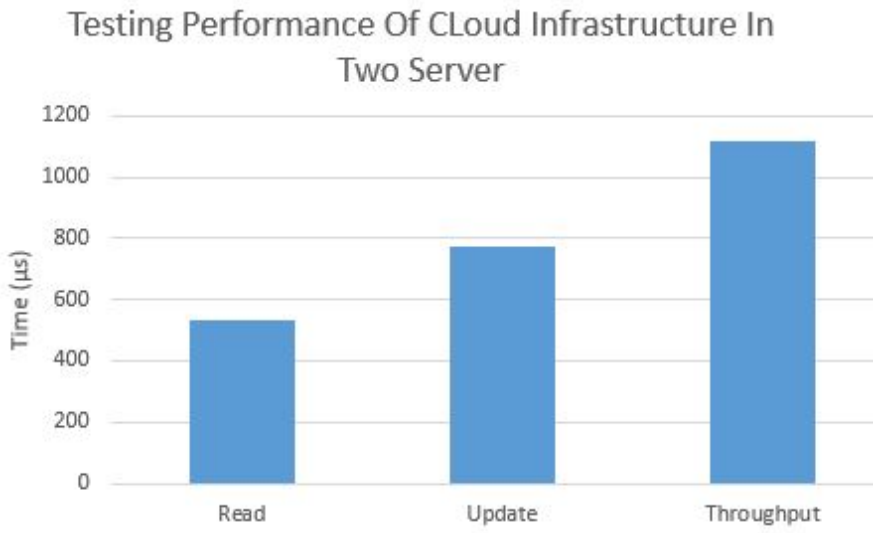

Figure 5.4: Test Performance Of Two Server Model

As we can see from the graph 5.4 the MongDB infrastructure is deploy in two different server. However, it is th deployment of the infrastructure is done in the same cloud environment. This deployment infrastructure is our proposed infrastructure which will show the improvement of the MongoDb. The comparison result of the two different models is show im the next part of this same chapter.

5.1.3 Comparing the performance of Cloud Infrastructure deployment between the Single server model vs Two server Model

In this section, we describe the comparison between the Single server model and the Two server Model by taking the average latency time in microseconds (μs) as shown in the table 5.3 and the graph 5.5

Table 5.3: Average Test Performance Single Server Vs Two Server

Test	**Server-1 (μs)**	**Server-2(μs)**
Read	2027	529
Update	2845	772
Throughput	429	1114

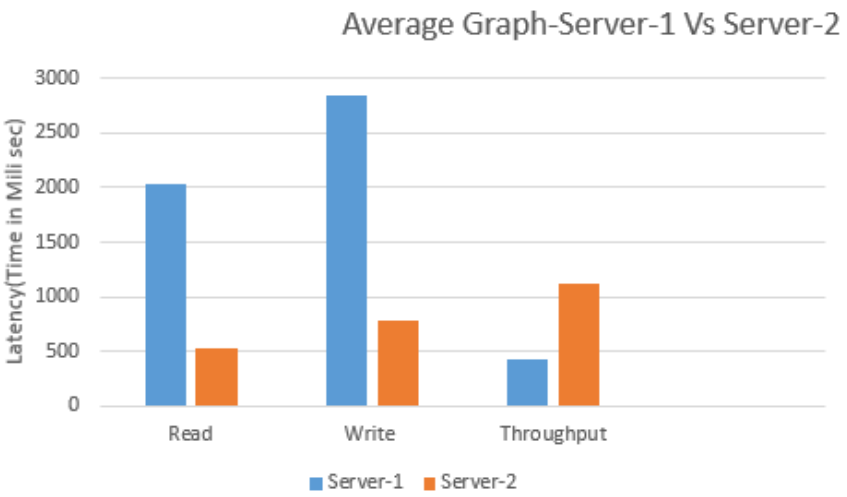

Figure 5.5: Average Test Performance-Server-1 Vs Server-2

The graph 5.5, shows the clear picture of the improvement of our proposed architecture when compare to the existing infrastructure of MongoDb. Our architecture brings numerous benefit to the organisation with the successful implementation of virtualization technology in our proposed cloud infrastructure. This brings benefits like low cost of resources, less resource utilisation and all resources are being used without making any of the servers idle as we can move the infrastructure with the availability of the resources. Our proposed infrastructure not only improve the read operation or the write operation, but it even brings improvement in the throughput of the data, therefore it depend on the individual to adopt our architecture because it have improvement in all the basic operations of the infrastructure.If any of the organisation make used of our proposed infrastructure they can take all the benefits which is posses by our infrastructure which is apply to any database they used. This is because when using the cloud environment in any organisation, they should make sure that the infrastructure of the database is chosen the best one. Since cloud infrastructure is the backbone of the cloud environment as we had mentioned in the previous chapter. The next part of this chapter we will show the testing performance when our proposed infrastructure is deploy in different cloud environment.

5.2 Deployment Performance Testing in Multiple Cloud Vendor

This section we will demonstrate the testing results of the Cloud infrastructure deployment in Multi Cloud Vendor. To test this we will deploy MongoDB clustered according

to our proposed model infrastructure, where we will use two server. In one server, we will deploy the router and Config,in the other server we will migrate the shard. Finally, we load the YCSB workload to check the latency rate of update, read and throughput calculated in microseconds(μs) .

5.2.1 AWS Cloud Vendor

In this section we deploy the MongoDB clustered architecture in AWS cloud vendor, in order to check the performance of the cloud infrastructure using two servers in the single cloud. We perform the testing 5 times as shown in the table 5.4 and the graph 5.6.

Table 5.4: Testing Performance Of Cloud Infrastructure Deployment In AWS

Operation	Latency s
Read	1013
Update	2845
Throughput	429.4

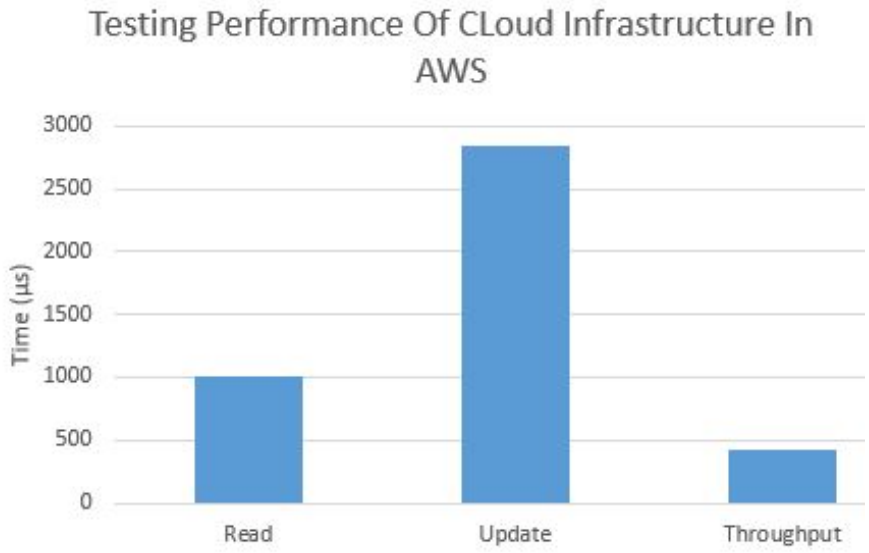

Figure 5.6: Testing Performance Of Cloud Infrastructure In AWS

5.2.2 Open Stack

We deploy the MongoDB clustered architecture in Openstack by running two servers and create 5 instances, where routers and config server is deployed in one server and

two shards are deploy in the other server. Finally, we test the cloud infrastructure deployment by loading the YCSB workload and generate the read,update and throughput latency in microseconds (μs). The test is performed 5 times as shown in the table5.5 and graph 5.7

Table 5.5: Testing Performance Of Cloud Infrastructure Deployment In Openstack

Operations	Latency s
Read	1461
Update	537
Throughput	643.8

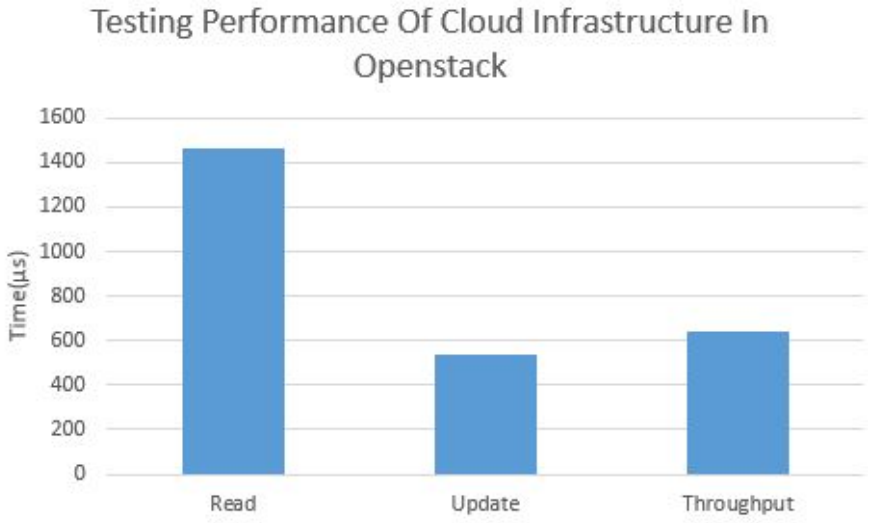

Figure 5.7: Testing Performance Of Cloud Infrastructure In Open Stack

The graph 5.7 shows the test result of the cloud infrastructure of MongoDB which has been run for 5 times. The result of each test does not have much different, but its stay consistence.

5.2.3 Digital Ocean

In this section, our proposed cloud infrastructure deployment is deployed in Digital Ocean. In which we create two servers and four containers and we deploy MongoDB clustered architecture over it. On one server, we deploy 2 container, i.e. config and router in the other sever we create 2 container that is two shards, to check the performance of the infrastructure in Digital ocean. The table 5.6 show the testing result

of the MongoDB clustered architecture, by generating the latency rate of read,update and throughput on loading the YCSB load.

Table 5.6: Testing Performance Of Cloud Infrastructure Deployment In Digital Ocean

Operations	Latency s
Read	996
Update	628.2
Throughput	782.8

The figure 5.8 shows the graphical representation of the performance of cloud infrastructure of MongoDB which is deployed in Digital ocean cloud platform with our proposed infrastructure.

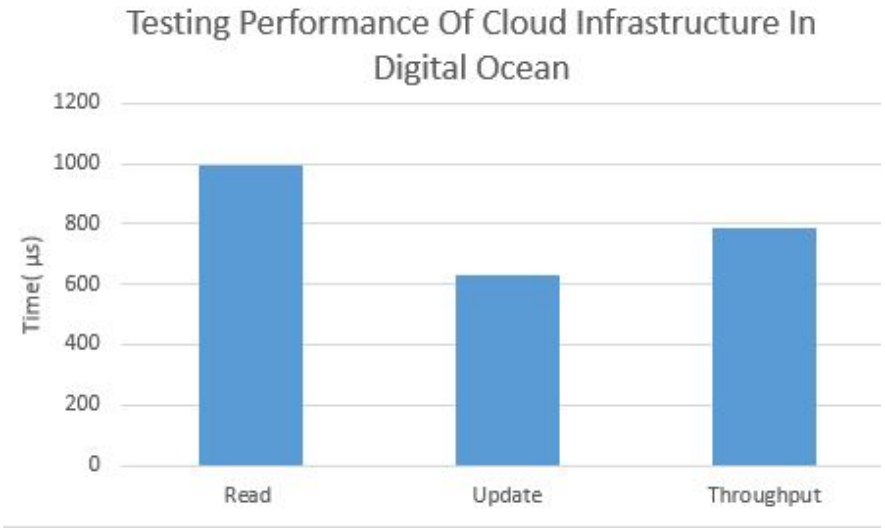

Figure 5.8: Testing Performance Of Cloud Infrastructure In Digital Ocean

5.2.4 Average Performance Of Cloud Infrastructure Deployment In Multiple Cloud Platform

In this section, we provide the average test result of the cloud infrastructure deployment of MongoDb architecture deployed in three different cloud platform. Every cloud vendor has their best services, however the cost differs from each other and the amount of resources they provide vary from each other. Our proposed infrastructure, after performing the test in three different clouds show the consistency of the latency between Digital Ocean and Openstack. To our recommendation,Openstack and Digital Ocean can be used by the organisation who worried about the cost and security. Because

Openstack provides high security services as it is the private cloud and Digital Ocean provides resources in lower price as compare to the other cloud providers.

Table 5.7: Test Performance Of cloud Infrastructure Deployment In Digital Ocean

Test	AWS	DigitalOcean	OpenStack
Read(μs)	1013	966	1461
Update(μs)	492	628	537
Throughput(μs)	605	783	643

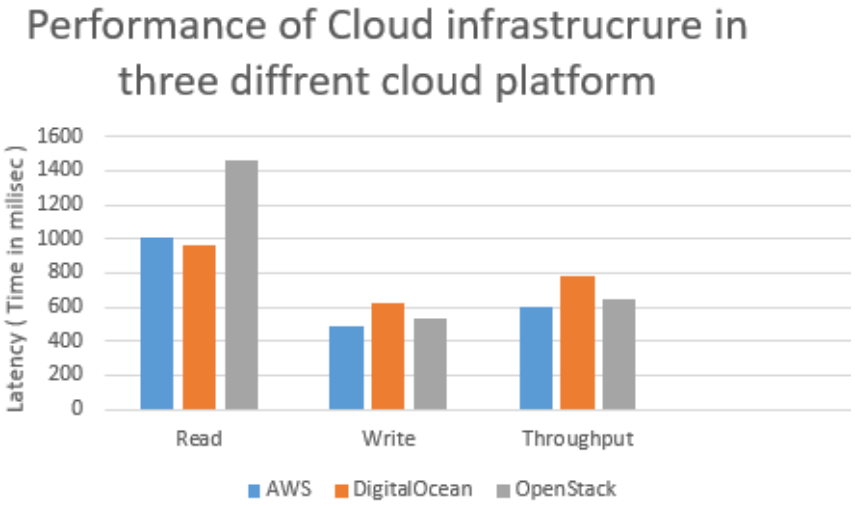

Figure 5.9: Performance Of Cloud Infrastructure Deploy In Three Different Cloud Infrastructure Platform.

As shown by the graph 5.9 our proposed architecture will bring various benefits to the organisation when it is adapted because the user can deploy and migrate the container anywhere in the cloud platform they needed. The user can implement the best cloud infrastructure at the very low cost. This is because we have implement virtulization technology in our proposed architecture. Our proposed architecture also improved the connectivity of Docker container with the external environment, which implies when the containers is connected with the multiple cloud platform. Our proposed infrastructure support the migration of data among different cloud without lost of any data. For instance, in case if one cloud vendor is under maintenance, or when the price of one vendor is high they can migrate the container to the other vendor which is lower in cost with more resource. As we seen from the graph 5.9 the average performance among three cloud vendor. In which Digital Ocean shows to be more beneficial to the

organisation when used by the organisation. We can see the read, update operation in Digital Ocean is better when compared with the other vendor. So this will bring more benefits to the organisation, since the cost of the Digital Ocean is more cheaper than the other vendor.It also provide more and better services with the low cost. The reason behind Digital Ocean to be improved in performance is because of the best type of SSD Harddrive they used. As we had mentioned in the previous chapter the we will be using the YCSB tools for testing the performance.

5.3 Resource Utilization:

In this section, we analyse the amount of resource consumed by our prototype and weave networking plug-in. Using an on-line monitoring tool called New Relic, we will analyse the resource utilisation of both the networking server into two different sections.Firstly, will analyse the CPU utilisation and secondly we will analyse the memory utilisation.

5.3.1 CPU utilisation:

In this section, we will evaluate the amount of CPU utilisation rate when the cluster is been deploy using weave networking plug-in and the DNS server. The figure 5.8 depicts CPU utilisation rate between Weave and DNS.

Table 5.8: CPU Utilisation Rate

	Docker	**Host**
DNS	1.80%	0.00%
Weave	1.64%	1.20%

From the graph 5.10 we could see that there is no value of DNS on host has been captured, the reason is that the DNS server in running inside the container whereas weave runs both in the docker and host machine. In this section we will evaluate the CPU utilized rate when the cluster is being deployed using weave networking plug-in and the DNS server architecture, this is analyse by using New Relic monitoring tool. The figure 5.10 depicts CPU utilisation rate between Weave and DNS. As we can see from the graph the Weave networking server make use of both Host and Docker it uses about 1.8% of the resources of Docker and 0.64% of the Host,whereas our DNS server does not consume any resources of the Host as it is created directly in the Docker.

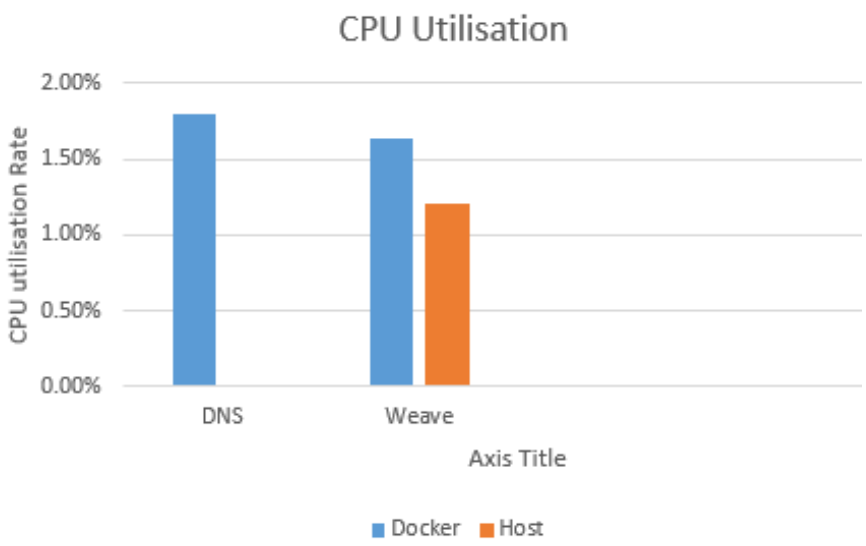

Figure 5.10: CPU Utilization Rate

5.3.2 Memory Utilization:

In our research we have used host machine with 512 MB memory. This memory will be utilized by all the containers on the host. The table 5.9 shows that the memory utilized by the DNS Configuration is less when compared with Weave.

Table 5.9: Memory Utilization.

	Host	**Docker**
Weave	42MB	26MB
DNS	–	28MB

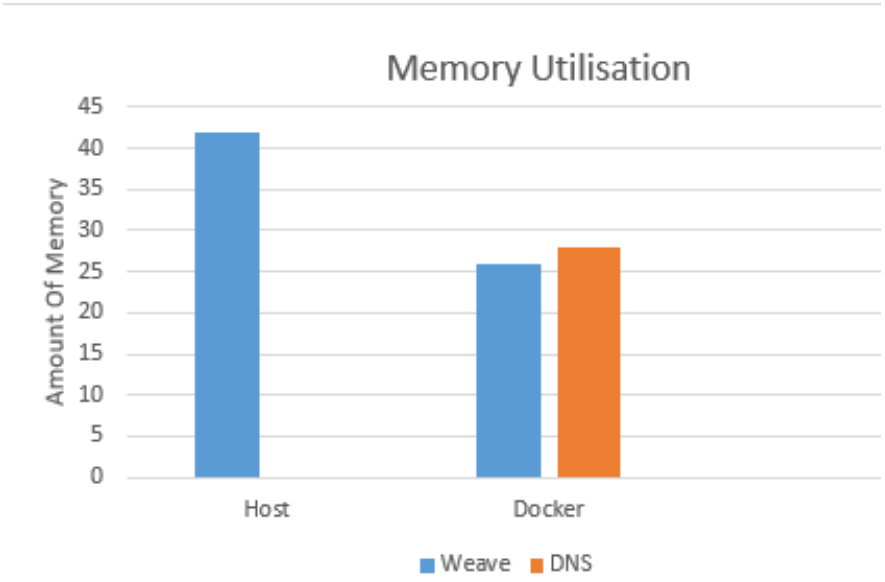

Figure 5.11: Memory Utilization.

As we can see from the figure 5.11, weave plug-in utilise more of the memory as compare to the DNS,this is because the weave are create over the Host and they even make use of the docker memory which mean its make used both physical and visualised memory resources. However, our proposed DNS server make use only of the vitalized resources which brings high benefits to the organisation if they adopt our proposed networking server connection technology because it will provide more resources in lower cost.

5.4 Bandwidth Availability.

It is important to maintain high bandwidth available between the servers in our infrastructure. The reason to check the bandwidth is that the docker networks docker0 bridge with the native host Ethernet interface. So there will be slight decrease in transfer rate to the outer world. The table5.10 below shows the amount of transfer rate and latency from one container to another container in a different host/network. We have captured the transfer rate from native to native, weave to weave and DNS to DNS by using Qpref monitoring tool.

1. **TCP transfer Rate:** This provides the speed of data required to transmit from one node to another node in network. The data transfer rate can be measure in (Mbps) megabits per second or mega byte per secong(Mbps).In another words, it is called as throughput, however in MongoDb it is term as number of operation perform by the system per second.

2. **TCP latency rate:** This provides the amount of time taken by the workload to transfer the data from one node to the other node. In our research, we generate the latency rate between three different networking connectivity weave to weave, DNS to DNS and native to native.

3. **Bandwidth rate:** Bandwidth rate is said to be the average rate of successful data transfer through a communication channel.

Table 5.10: Bandwidth Availability

	Native	**Weave**	**DNS**
TCP(MBs)	330	120	226
TCP lat.(μs)	96.2	372	236
BW(%)	100	9.09	79.32

The figure 5.12 depicts the transfer rate between the containers we generate the transfer rate from native to native, weave to weave and DNS to DNS using TCP protocol.

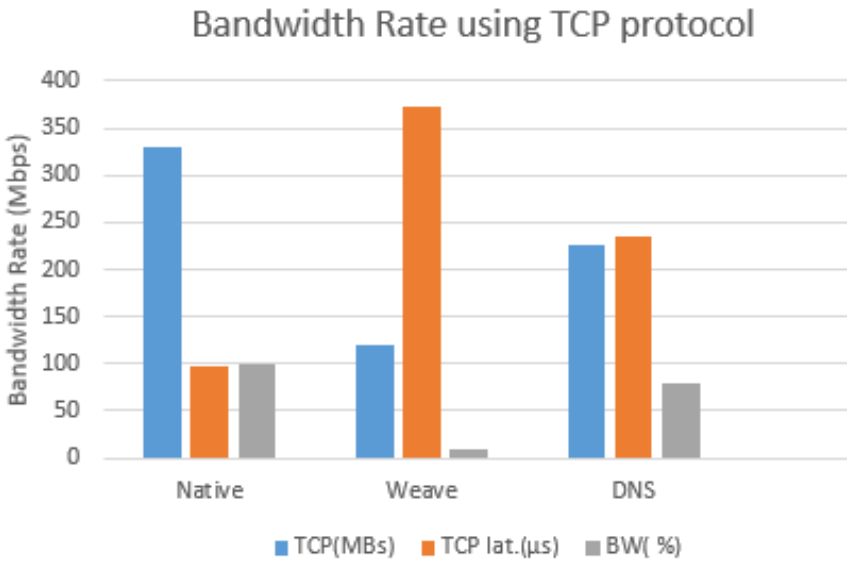

Figure 5.12: Transfer Rate Between containers.

The graph 5.12 shows the Bandwidth rate using TCP protocol. When the data is transfer using Weave networking connectivity and DNS connectivity, we can see that DNS data transfer bandwidth is better in compare to WEAVE plug-in. Weave plug-in have its disadvantages in which the services need to be shut down when infrastructure need to increase or improved. However, when the DNS connectivity is used we can

increase the infrastructure without any shutting down of the server. This show that our proposed model of implementing the DNS connectivity in our project overcome the problem of connectivity in Docker Container.

Chapter 6

Conclusion

Hypervisor based virtualization technique are not comparable for multiple cloud environment. Hardware requirements and resource allocation strategies are different in Docker based cloud infrastructure. So here we conclude the need of Docker based virtualization solution in cloud infrastructure. This infrastructure deployment in multiple cloud platform will achieve high availability. This thesis endorses a Docker based cloud infrastructure solution for new generation multiple cloud vendors. In our research we have also addressed the challenges faced by Docker networking that occur during the cloud infrastructure deployment described in implementation section, we have provided the solution in design section by creating a private DNS server in each cloud platform and the container are connecting to each other through the private DNS.

The test result of our implementation section is generated and describe in section 5.In which we deploy the MongoDB clustered architecture in the single server and we check the performance of the infrastructure. Similarly, we deploy the MongoDB architecture into our proposed architecture and compare the performance between the two model.The result that we can see from the evaluation chapter our proposed infrastructure improved the read and update operation of the MongoDB when compare to its core architecture. We also deploy our proposed infrastructure in Multiple cloud model out of which results to show the better performance of the infrastructure when it is deploy in Digital Ocean. We have also performed various test parameter in our research such, as resource utilisation where the result shows the low resources utilisation of the DNS connectivity server which is the core design of our research when compare to the Wesve plugin. Furthermost,using QPREF we check the Bandwidth availability of our proposed Cloud infrastructure deployment performance in multiple cloud and we have achieved our expected result.

6.0.1 Limitation and Future Work

In our research due to less available of resources and our prototype cant be extended. Docker based cloud infrastructure is benefits lots of innovation our research can still be continued further for the following approaches

On-line Portal:Building On-line dashboard for cloud bursting across multiple cloud platform.
Docker Swarm: This approach can still configure by using Docker swarm, which is a cluster monitor tool by Docker.

Our research can also be extended by deploying our proposed Cloud infrastructure in another cloud vendor like Widow Azure,JoyNet Cloud,IBM and many other big providers. The private DNS server can always act as the connectivity when tested the performance of the infrastructure no matter what database we use.Furthermost,without any concern on how many cloud environment we deploy our infrastructure we can carry on the implementation of new infrastructure without shutting down of the services.

Bibliography

Ahmad, N., Kanwal, A. and Shibli, M. A. (2013), Survey on secure live virtual machine (vm) migration in cloud, *in* 'Information Assurance (NCIA), 2013 2nd National Conference on', IEEE, pp. 101–106.

Anala, M., Kashyap, M. and Shobha, G. (2013), Application performance analysis during live migration of virtual machines, *in* 'Advance Computing Conference (IACC), 2013 IEEE 3rd International', IEEE, pp. 366–372.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. et al. (2010), 'A view of cloud computing', *Communications of the ACM* **53**(4), 50–58.

Bacis, E., Mutti, S., Capelli, S. and Paraboschi, S. (2015), Dockerpolicymodules: mandatory access control for docker containers, *in* 'Communications and Network Security (CNS), 2015 IEEE Conference on', IEEE, pp. 749–750.

Bernstein, D. (2014), 'Containers and cloud: From lxc to docker to kubernetes', *IEEE Cloud Computing* (3), 81–84.

Boettiger, C. (2015), 'An introduction to docker for reproducible research', *ACM SIGOPS Operating Systems Review* **49**(1), 71–79.

Burson-Marsteller and Starch, R. (2016), Application Performancde Monitoring APM for LAB, Technical report, New Relic Organisation.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J. and Brandic, I. (2009), 'Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility', *Future Generation computer systems* **25**(6), 599–616.

Casoni, M., Grazia, C. A. and Patriciello, N. (2013), On the performance of linux container with netmap/vale for networks virtualization, *in* 'Networks (ICON), 2013 19th IEEE International Conference on', IEEE, pp. 1–6.

Chandramouli, R. and Rose, S. (2005), An integrity verification scheme for dns zone file based on security impact analysis, *in* '21st Annual Computer Security Applications Conference (ACSAC'05)', pp. 10 pp.–321.

Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R. and Sears, R. (2010), Benchmarking cloud serving systems with ycsb, *in* 'Proceedings of the 1st ACM symposium on Cloud computing', ACM, pp. 143–154.

Deb, S., Srinivasan, A. and Pavan, S. K. (2008), An improved dns server selection algorithm for faster lookups, *in* 'Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on', pp. 288–295.

Dhungana, R. D., Mohammad, A., Sharma, A. and Schoen, I. (2013), Identity management framework for cloud networking infrastructure, *in* 'Innovations in Information Technology (IIT), 2013 9th International Conference on', IEEE, pp. 13–17.

Dong, Y., Yang, X., Li, J., Liao, G., Tian, K. and Guan, H. (2012), 'High performance network virtualization with sr-iov', *Journal of Parallel and Distributed Computing* **72**(11), 1471–1480.

Dong, Y., Zhang, X., Dai, J. and Guan, H. (2014), 'Hyvi: A hybrid virtualization solution balancing performance and manageability', *Parallel and Distributed Systems, IEEE Transactions on* **25**(9), 2332–2341.

Dua, R., Raja, A. R. and Kakadia, D. (2014), Virtualization vs containerization to support paas, *in* 'Cloud Engineering (IC2E), 2014 IEEE International Conference on', IEEE, pp. 610–614.

Dusia, A., Yang, Y. and Taufer, M. (2015), Network quality of service in docker containers, *in* 'Cluster Computing (CLUSTER), 2015 IEEE International Conference on', IEEE, pp. 527–528.

Fakhfakh, M., Cherkaoui, O., Bedhiaf, I. L. and Frikha, M. (2009), High availability in ims virtualized network, *in* 'Communications and Networking, 2009. ComNet 2009. First International Conference on', IEEE, pp. 1–6.

Felter, W., Ferreira, A., Rajamony, R. and Rubio, J. (2015), An updated performance comparison of virtual machines and linux containers, *in* 'Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On', IEEE, pp. 171–172.

Ismail, B. I., Mostajeran Goortani, E., Ab Karim, M. B., Ming Tat, W., Setapa, S., Luke, J. Y. and Hong Hoe, O. (2015), Evaluation of docker as edge computing platform, *in* 'Open Systems (ICOS), 2015 IEEE Confernece on', IEEE, pp. 130–135.

Jalalzai, M. H., Shahid, W. B. and Iqbal, M. M. W. (2015), Dns security challenges and best practices to deploy secure dns with digital signatures, *in* '2015 12th International Bhurban Conference on Applied Sciences and Technology (IBCAST)', pp. 280–285.

Jin, X., Wang, H., Wang, J., Cheng, S. and Li, J. (2013), A partners assisted virtual machine live storage migration for intensive disk i/o workloads, *in* 'High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on', IEEE, pp. 1693–1698.

Lei, S., Zishan, D. and Jindi, G. (2010), Research on key management infrastructure in cloud computing environment, *in* 'Grid and Cooperative Computing (GCC), 2010 9th International Conference on', IEEE, pp. 404–407.

Liu, C. and Albitz, P. (2006), *DNS and Bind*, " O'Reilly Media, Inc.".

Liu, D. and Zhao, L. (2014), The research and implementation of cloud computing platform based on docker, *in* 'Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2014 11th International Computer Conference on', IEEE, pp. 475–478.

Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L. and Leaf, D. (2011), 'Nist cloud computing reference architecture', *NIST special publication* **500**, 292.

Liu, X. X., Qiu, J. and Zhang, J. M. (2014), High availability benchmarking for cloud management infrastructure, *in* 'Service Sciences (ICSS), 2014 International Conference on', IEEE, pp. 163–168.

Liu, Y., Wang, Y. and Jin, Y. (2012), Research on the improvement of mongodb auto-sharding in cloud environment, *in* 'Computer Science Education (ICCSE), 2012 7th International Conference on', pp. 851–854.

Marquezan, C. C., Bruneo, D., Longo, F., Wessling, F., Metzger, A. and Puliafito, A. (2014), 3-d cloud monitoring: Enabling effective cloud infrastructure and application management, *in* 'Network and Service Management (CNSM), 2014 10th International Conference on', IEEE, pp. 55–63.

Mattetti, M., Shulman-Peleg, A., Allouche, Y., Corradi, A., Dolev, S. and Foschini, L. (2015), Securing the infrastructure and the workloads of linux containers, *in* 'Communications and Network Security (CNS), 2015 IEEE Conference on', IEEE, pp. 559–567.

Patnaik, D., Bijlani, A. and Singh, V. K. (2010), Towards high-availability for ip telephony using virtual machines, *in* 'Internet Multimedia Services Architecture and Application (IMSAA), 2010 IEEE 4th International Conference on', IEEE, pp. 1–6.

Raho, M., Spyridakis, A., Paolino, M. and Raho, D. (2015), Kvm, xen and docker: A performance analysis for arm based nfv and cloud computing, *in* 'Information, Electronic and Electrical Engineering (AIEEE), 2015 IEEE 3rd Workshop on Advances in', IEEE, pp. 1–8.

Rey, J., Cogorno, M., Nesmachnow, S. and Steffenel, L. A. (2015), Efficient prototyping of fault tolerant map-reduce applications with docker-hadoop, *in* 'Cloud Engineering (IC2E), 2015 IEEE International Conference on', IEEE, pp. 369–376.

Satam, P., Alipour, H., Al-Nashif, Y. and Hariri, S. (2015), Dns-ids: Securing dns in the cloud era, *in* 'Cloud and Autonomic Computing (ICCAC), 2015 International Conference on', pp. 296–301.

Sun, R., Yang, J. and He, Z. (2013), An approach to minimizing downtime induced by taking live snapshot of virtual cluster, *in* 'Cloud and Service Computing (CSC), 2013 International Conference on', IEEE, pp. 63–68.

To, M. A., Cano, M. and Biba, P. (2015), Dockemu–a network emulation tool, *in* 'Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on', IEEE, pp. 593–598.

Yan, S., Lee, B. S., Zhao, G., Ma, D. and Mohamed, P. (2011), Infrastructure management of hybrid cloud for enterprise users, *in* 'Systems and Virtualization Management (SVM), 2011 5th International DMTF Academic Alliance Workshop on', IEEE, pp. 1–6.

Yu, C. and Huan, F. (2015), 'Live migration of docker containers through logging and replay'.