DISASTER RECOVERY MECHANISM WITH ACTIVE STORAGE REPLICATION AND DNS FAILOVER IN CLOUD VIRTUAL MACHINE

A PROJECT REPORT

Submitted by

BALASA CHENCHU SAI VYSHNAVI	**211418104036**
KOLLA PRIYA	**211418104129**
KURICHETI SPOORTHI	**211418104136**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MAY 2022

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"DISASTER RECOVERY MECHANISM WITH ACTIVE STORAGE REPLICATION AND DNS FAILOVER IN CLOUD VIRTUAL MACHINE"** is the bonafide work of **"BALASA CHENCHU SAI VYSHNAVI [REG NO: 211418104036], KOLLA PRIYA [REG NO: 211418104129] and KURICHETI SPOORTHI [REG NO: 211418104136]"** who carried out the project work under my supervision.

SIGNATURE

Dr.S.MURUGAVALLI,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
CHENNAI-600 123.

SIGNATURE

Mrs. DEVI .R, M.E.,
SUPERVISOR
ASSOCIATE PROFESSOR

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE
CHENNAI-600 123.

Certified that the above mentioned students were examined in End Semester project viva voice held on _____.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We BALASA CHENCHU SAI VYSHNAVI[211418104036], KOLLA PRIYA [211418104129] and KURICHETI SPOORTHI [211418104136] here by declare that this project report titled "DISASTER RECOVERY MECHANISM WITH ACTIVE STORAGE REPLICATION AND DNS FAILOVER IN CLOUD VIRTUAL MACHINE", under the guidance of Mrs. DEVI R, M.E., is the originalwork done by us and we have not plagiarized or submitted to any other degree in any university by us.

BALASA CHENCHU SAI VYSHNAVI

KOLLA PRIYA

KURICHETI SPOORTHI

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to express our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHIKUMAR, M.E., Ph.D.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E., M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI, M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my Project Guide, **Mrs. DEVI R, M.E.,** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

B.C.SAI VYSHNAVI
KOLLA PRIYA
KURICHETI SPOORTHI

ABSTRACT

The Domain Name Service (DNS) is a vital service on the Internet. It allows for higher-profile functionalities like load balancing and improved content distribution, in addition to simple translation.DNS is envisioned as an elastic and reliable service in the cloud, supporting failover mechanisms, decentralized configuration, and multi-tenant isolation. The problem arises where a particular DNS is not able to point an IP address of the server because of heavy traffic, DDoS attack, system crash and much more reasons such as if multiple request is sent to the server then the server cant handle everything at same this leads to network traffic so the crash in server takes place. During this period there are huge chances for an organization to be a victim of huge, valuable data loss and have downtime for a certain period. This will cause a billion dollar business impact and risk of losing data and loss of network connectivity so may not be able to find the information or carry out the actions they need. To overcome this we come up with an approach of DNS Failover with active Replication which is a solution designed to help keep the services online and prevent the servers from Data losses. When systems and services go down DNS failover is used to direct the users to another resource with little to no disruption which has active replication of Data with the current prod Server ie., a database contains two servers if one is failed then another will work with same data without any data loss. The backup server/Replicating server is named as Contingency server (CNR) which has an active storage replication with the Production server preventing Data loss and the other server is backup server which will replicate the data from the active server when any updation takes place in the active server.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

S. NO.	ABBREVATION	EXPANSION
1	DNS	Domain Name System
2	IP	Internet Protocol
3	BIND	Berkeley Internet Name Domain
4	CNR	Contingency Server
5	NAS	Network Attached Storage
6	VM	Virtual Machine
7	EFS	Elastic File System
8	SCP	Secure Copy Protocol
9	DDOS	Distributed Denial of Service
10	MORM	Mask Read Only memory
11	CSO	Compressed ISO file
12	CRUD	Create, Read, Update and Delete
13	AD	Azure Directory
14	ISCSI	Internet Small Computer Systems Interface
15	EC2	Elastic Cloud services
16	AZ	Azure Web Services
17	SAN	Storage Attached Network
18	EFS	Elastic File System
19	FDNS	Forward Domain Name Service

CHAPTER 1
INTRODUCTION

1.INTRODUCTION

1.1 OVERVIEW

The Domain Name System (DNS) is a critical and fundamental Internet service that maps domain names to IP addresses. Web servers of campus networks provide external services by two means: private network address and mapping public network address. Summarizing, the last one means that the internal users access the private address through high-speed internal network, while the external users adopt the public network route to access the public network address which has been translated. The campus network DNS server is usually divided according to the scope of customer address, returning different addresses to resolve private addresses for internal users, and to resolve the public network address for internet users. The most widely known DNS server is Berkeley Internet Name Daemon (BIND). The function "View" BIND9 can realize that different source IP ranges succeed in gaining different response information. The methods of policy DNS can provide specific customer scope with corresponding network address resolution, although some problems may appear during this process. If the network with multiple operator exits and the IP address in the DNS server resolution fails to use the particular route in the routing route selection, or fails to find corresponding route entry, this will lead to slower access speed even the access failure. To handle the above mentioned problems we come up with an approach of DNS Failover with active Replication which is a solution designed to help keep the services online and prevent from Data losses. When systems and services go down DNS failover is used to direct the users to another resource with little to no disruption which has active replication of Data with the current prod Server i.e here we take 2 servers one is set as active servers and another as backup server. The backup server/ Replicating server is named as Contingency server (CNR) which has a active storage replication with the Production server preventing Data loss i.e., when the user updates any data in active server the data are replicated to backup server so when the active server is crashed then the backup server will work with the same data. To direct

the traffic to another resource with less traffic we use concept of Forward Domain Name Service (FDNS) controlled with Apache KNOX to perform Automated DNS failover to less traffic backup server.Forward DNS is a type of DNS request in which a domain name is used to obtain its corresponding IP address . To avoid risk of losing production of valuable data we use virtual shared disk storage attached to cloud VM having Network Attached Storage (NAS) which uses SCP and Elastic File system (EFS) to transfer Data back and front to the Cloud storage disk. We believe using the following method servers can be prevented from DDoS attack and can have an auto recovery mechanism which eliminates service downtime. For more security a SSH key handshake mechanism will be implemented in the server to prevent Trespassers to override the Production Data.

1.2 PROBLEM DEFINITION

To ensure continuous performance and minimize service downtime, services can be switched from the active server of a high-availability cluster to the redundant, passive server. This feature is known as "Switch-over" or "Failover" in some circumstances. During this period there are huge chances for an organization to be a victim of huge, valuable data loss and have a downtime for a certain period.

CHAPTER 2

LITERATURE SURVEY

2.LITERATURE SURVEY

Disaster recovery mechanism in cloud has currently received a lot of attention. A handful of research has been done on data replication algorithm and cloud environment. In [1] the author proposed an algorithm, namely RSPC, that introduced a replication strategy that fulfills the inhabitant targets and the supplier benefit. It demonstrates a popularity based file selection strategy through analyzing data access by users in a determined period of time.

Mansouri, et al [2] addressed the approach for secure data replication. They proposed the Secure Data Replication (SDR) algorithm, which identifies popular files as replication candidates based on the number of file-access times, where the higher the number of file-access times, the higher the popularity. Following that, only 20% of the candidates will be chosen for replication. Moreover, to improve the performance in cloud, they also proposed [3] a dynamic replication algorithm namely Hierarchical Data Replication Strategy, that identifies the popularity of a file based on a predicted number of file-access times then it replicates that popular file into the best site using network level locality.

In paper [4], Yanling Shao, et al presented a replication management system that is cost effective and reliable. It is implemented utilising the Dynamic Replica Creation Algorithm (DRCA) and Data Replica Scheduling Algorithm (DRSA), which are based on a meta-heuristic method in which files are selected depending on their popularity. This improves system's performance and allows for the search of a higher-quality replica placement solution while also lowering total data access costs while meeting the deadline.

Xiong Fu, et al [5] used Replica Placement Based on Load Balance strategy (RPBLB) to reduce remote users access time. RPBLB recommended replicating the most frequently used data to new storage while keeping storage load balancing in mind. This also ensures the load balance between data centers.

A.U. Rehman, et al [6] in his work, addressed the Software Defined Networking (SDN) fault-tolerance and discussed the OpenFlow fault-tolerance support for failure recovery. They proposed the make function such as firewall and Domain Name Service (DNS) to their customers. They further highlighted the SDN-specific fault-tolerance issues and provide a comprehensive overview of the state-of-the-art SDN fault tolerance research efforts.

S. Zheng, et al [7] presented Ziggurat, a multi-tiered file system that combines main memory which is non-volatile and slow disks to create a storage system with near-NVMM (Non Volatile Main Memory) performance and large capacity. Depending on application access patterns, write size, and the likelihood that the application will stall until the write completes, Ziggurat routes incoming writes to NVMM, DRAM, or disk. They proposed Emerging fast, byte-addressable Non-Volatile Main Memory (NVMM) that provides huge increases in storage performance compared to traditional disks.

V. Bindhu [8] the author in her paper proposed zero-downtime deployment, would ideally not cause any outage to the end users. The old version continues to run till the new version is ready. Perhaps the most strong innovation in improving the transmission capacity usage of the next generation network is cognitive radio network (CR-N). Anyway, the conventional CR-N is significantly compelled in getting to and the range detecting, because of its restricted, handling power and the stockpiling capacities.

Ernesto Garbarino [9] provided a guide to automating application deployment, scaling and management. This helps to learn and set up Kubernetes and achieve zero-downtime deployments using the service controllers. The combination of micro service architecture, containerization, orchestration and public cloud compute resources have significantly reduced the time-lines. However, the modern businesses are extremely dynamic and fluid in nature. They demand changes in the web applications continuously and rapidly.

Jayashree Mohan, et al [10] presented CheckFreq, which is an automatic, fine grained checkpointing framework that algorithmically determines check pointing frequency at the granularity of iterations using online monitoring. Moreover, it carefully pipelines check-pointing with computation to reduce the checkpoint cost by introducing two-phase check-pointing. They investigated various check pointing approaches such as asynchronous check pointing in High Performance Computing, and among those approaches, synchronous check pointing occurred large checkpoint stalls.

CHAPTER 3
SYSTEM ANALYSIS

3.SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In today's world, new data-intensive applications can benefit if cloud schedulers use data replication strategies in executing their workflows. In this environment, storing replicas at multiple data centers and then accessing them from the appropriate data center provides efficient data access without a large consumption of bandwidth. Due to the high bandwidth requirements, replicating all files across all data centres is not feasible. Also, data centers have not always had enough space to store all these data files. In order to address these issues, it duplicates vital data to the most appropriate areas. Many academics have worked on data replication in a cloud context to achieve a satisfactory QoS. Multi-objective Optimized Replication Management (MORM) strategy by balancing the trade-offs among the different parameters in a cloud environment. They proposed a formula that is a combination of various optimization factors such as data availability, load, latency, service time, energy consumption, probability of failure. Users can set the weight of variables based on the requirements. For example, assigning a higher value on their performance objectives makes the replication strategy adaptable. The experimental results using the extended CloudSim showed that MORM strategy can enhance load balancing of the system and reduce energy consumption. But they did not deal with the dynamic variation in file access characteristics. Furthermore, this project does not cover in-depth application analysis.

In Existing system the above problems are solved using the following methods: DNS **QUERY:** The DNS client contacts a DNS resolver to request an IP address for that host name. The DNS resolver tries to locate the DNS server that holds the correct IP address and resolves the query by returning the DNS record to the client.

DOWNSIDE: Time consuming and has a failover delay beyond 1hr Authoritative Name Server: A DNS resolver starts resolving a query by looking at its local cache for the required IP address or name servers of the required host, failing that it performs a

recursive search with the request query starting from the internet root DNS **DOWNSIDE:** Time consuming and has no storage replication SAN/NAS based Storage replication: This performs storage replication and as well as Wide ip failover but its manual failover is possible and has a delay of 30mns to 1hr and storage replication is not real time which leads a risk of loosing the data for the time frame.

3.2 PROPOSED SYSTEM

There has been a few research that applies data replication to reduce energy consumption, transfer time as well as achieving data security in a cloud environment. Most of the service providers like Amazon S3 use the security schema and encrypt the file based on the security level, and so significant delay has appeared.We propose a new data replication strategy for cloud computing which offers data security while minimizing the overhead of the security service by considering important factors and concepts.

The major contributions can be listed as follows.

• In a cloud environment, simulate the data center's energy consumption during data replication.

• Consider the centrality measure to improve retrieval time.

• Assign merit value to all data centers by designing a fuzzy inference system with four inputs as load, storage usage, energy, and centrality. When control processes and decisions are difficult to analyse using traditional quantitative methods, a fuzzy inference system comes in handy.

• High-quality fuzzy rules are determined by the CSO algorithm in the design of a flexible fuzzy system.

• Develop a scheme for data partitioning and take into account the data center suitability in fragment size determination.

• To prevent an attacker from determining the locations of the fragments, the selected data centres are separated using the T-coloring approach. Therefore, the proposed

algorithm addresses achieve fast data retrieval while guaranteeing the security of data. Let there are K data centers in the cloud, and a file is divided into n fragments. Consider the number of successful intrusions on a data center is m, such that m > n. P(m,n) indicates the probability that m number of victim data centers have all of the n data centers containing the fragments of a file.

To handle the above mentioned problems we come up with an approach of DNS Failover with active Replication which is a solution designed to help keep the services online and prevent from Data losses. When systems and services go down DNS failover is used to direct the users to another resource with little to no disruption which has active replication of Data with the current production Server. The backup server/ Replicating server is named as Contingency server (CNR) which has an active storage replication with the Production server preventing Data loss. To direct the traffic to another resource with less traffic we use the concept of Forward Domain Name Service (FDNS) controlled with Apache KNOX to perform Automated DNS failover to less traffic backup server. To avoid risk of losing production of valuable data we use virtual shared disk storage attached to cloud VM having Network Attached Storage (NAS) which uses SCP and Elastic File system (EFS) to transfer Data back and front to the Cloud storage disk. We believe using the following method servers can be prevented from DDoS attack and can have an auto recovery mechanism which eliminates service downtime. For more security a SSH key handshake mechanism will be implemented in the server to prevent Trespassers to override the Production Data.

3.3 FEASIBILITY STUDY

1. Helps websites or network services remain accessible in the event of outage.
2. Obtains the IP address or other required details and resolves the query, by returning the DNS record to the client.
3. Replication offers disaster recovery and preparedness capabilities in Windows.

4. Storage Replica may allow you to decommission existing file replication systems such as DFS Replication that were pressed into duty as low-end disaster recovery solutions.

AREAS OF FEASIBILITY

3.3.1 SOCIAL FEASIBILITY:

- Reduces the screen timing of the users by decreasing the failover delay
- prevention of the data loss,so the data required for the end users is safe and secure

3.3.2 ECONOMIC FEASIBILITY:

- The financial cost related to this project it feasible as it uses cloud, and open sources like postgresql database

3.3.3 OPERATIONAL FEASIBILITY:

- It will be highly useful for the people so the operational system is feasible and it is also way more better than the existing system

3.3.4 SCHEDULE FEASIBILITY:

- Based on the designed timeline chart the proposed system only requires 2-3 months for developing it without any delay.

3.3.5 TECHNICAL FEASIBILITY:

PYTHON:

- Since this is an artificial intelligence project using python gives us the advantage for various inbuilt libraries like TensorFlow, Scikit-Learn, and NumPy.
- Python is an object-oriented programming language that makes coding easier and less prone to syntax errors.
- Python being platform-independent will help in future up-gradation to deploy as software.

POSTGRESQL:

- POSTGRESQL is best suited for systems that require execution of complex queries and data analysis

GUNICORN:

- Green Unicorn is a python web server.
- The Gunicorn server is widely compatible with a variety of web frameworks, is easy to set up, uses minimal server resources, and is relatively quick.

3.4 REQUIREMENT ANALYSIS :

System Requirement Specifications (SRS)

A System Requirement Specification (SRS) is basically an organization's understanding of a customer or potential client's system requirements and dependencies at a particular point prior to any actual design or development work. The information that is gathered during the analysis is translated into a document that

defines a set of requirements. It gives a brief description of the services that the system should provide and also the constraints under which the system should operate. Generally, SRS is a document that which completely describes what the proposed software should do without explaining how the software will do it. It's a two-way insurance policy that assures that at a given point of time both the client and the organization understand the other's requirements from that perspective . SRS document itself states in precise and explicit language those functions and capabilities a software system (i.e., a software application, an ecommerce website and so on) must provide, and also mentions any required constraints by which the system must abide. SRS also works as a blueprint for finishing a project with as little cost growth as possible. SRS is usually identified as the "parent" document because all subsequent project management documents, like design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are associated to it.As we know that requirement is a condition or capability to which the system must conform. Requirement Management is a systematic approach towards eliciting, organizing and documenting the requirements of the system properly along with the applicable attributes. The elusive difficulties of requirements are not always obvious and might come from any number of sources.

PURPOSE:

To ensure continuous performance and minimize service downtime, services will switch from the active server of a high-availability cluster to the redundant, passive server. This feature is called "Switchover," or in some cases "Failover." During this period there are huge chances for an organization to be a victim of huge, valuable data loss and have downtime for a certain period. This will cause a billion dollar business impact and risk of losing data.

3.4.1 Functional Requirement:

Functional Requirement defines the function of a software system and also how the system must behave when presented with specific inputs or conditions.These may include calculations, data manipulation and processing and other specific functionality.-
Following are the functional requirements on the system:

1. The entire control model set must be translated to Python and Shell script output Code.
2. Inputs must be Server ip, dns name and basic CRUD operation along with standard design components
3. Multiple servers and Storage units must be processed and the result must be combined to obtain a single output file.

3.4.2 Non-Functional Requirement:

Nonfunctional requirements are those which are not directly concerned with the specific function delivered by the system. They specify the criteria that could be used to judge the operation of a system rather than specific behaviors. They might relate to emergent system properties like reliability, response time and store occupancy. Nonfunctional requirements arise through the user needs , organizational policies ,due to budget constraints, the necessity for interoperability with other software and hardware systems or due to external factors such as: -

- ➢ Product Requirements
- ➢ Organizational Requirements
- ➢ User Requirements
- ➢ Basic Operational Requirements

3.4.2.1 Product Requirements

Platform Independency: Standalone executables for embedded systems can be created so that the algorithm developed using available products could be downloaded on the actual hardware and executed without any dependency to the development and modeling platform.

Correctness: This follows a well-defined set of procedures and rules to compute and also rigorous testing is done to confirm the correctness of the data.
Ease of Use: Model Coder will provide an interface which allows the user to interact in an easy manner.

Modularity: The complete product is broken up into numerous modules and well-defined interfaces are developed to explore the benefit of flexibility of the product.

Robustness: This software is being developed in such a way that the overall performance is optimized and the user can expect the results within a limited time with utmost relevance and correctness Nonfunctional requirements are also called the qualities of a system. These qualities can be distinguished into execution quality & evolution quality. Execution qualities are those which are observed during the run time such as security & usability of the system. whereas testability, maintainability, extensibility or scalability comes under evolution quality.

3.4.2.2 Organizational Requirements

Process Standards: The standards defined by AZURE are used to develop the application which is the standard used by the developers inside the defense organization.

Design Methods: Design is one of the most important stages in the software engineering process. This is the first step in moving from problem to the solution domain. In other words, starting with what is needed, design will take us to work on how to satisfy the needs.

3.4.2.3 User Requirements

- The coder must request the name of the server and Disk to be processed
  1. A web application where one could test the server activeness and DNS failover with storage replication
  2. Azure AD portal for active server switching and schedule maintenance
  3. Control replication server and Automate DNS failover with Storage replication with web application
- In case of multiple files, the coder must ask the names of the files in sequential manner.
- The output file must be a Python or shell script translated from the model.
- Even if multiple input files are provided only a single output file must be created.

3.4.2.4 Basic Operational Requirements

The customers are the ones that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, those will be related to these below points: -

Mission profile or scenario: This explains the procedures that are used to accomplish mission objectives,and also finds out the effectiveness or efficiency of the system. Performance and related parameters: It points out the critical system parameters in order to accomplish the mission

Utilization environments: This gives a brief outline of system usage. Finds out suitable

environments for effective system operation. Operational life cycle: It defines the system lifetime.

3.5 HARDWARE ENVIRONMENT

Processor	-	Azure cloud processor > D series
Speed	-	1.1 Ghz
RAM	-	4GB RAM
Hard Disk	-	30 GB
Key Board	-	US/ UK
Mouse	-	Two or Three Button Mouse
Monitor	-	Virtual Machine

- Processor : core i5/17/i8
- RAM : 4GB
- CPU : 2 x 64-bit 2.8GHZ 8.00 GT/s
- Disk storage : 500 GB.
- NAS based Virtual Storage Disk (AZURE)

3.6 SOFTWARE ENVIRONMENT:

Operating System	-	Unix/Linux
Coding Language	-	Python >= 3.8.0

- Operating system : Windows 7/8/10(64bit) / Unix/ Linux
- Additional : Shell Scripting, Django, python, Postgresql, Azure VM

CHAPTER 4
SYSTEM DESIGN

4.SYSTEM DESIGN

4.1 ER DIAGRAM

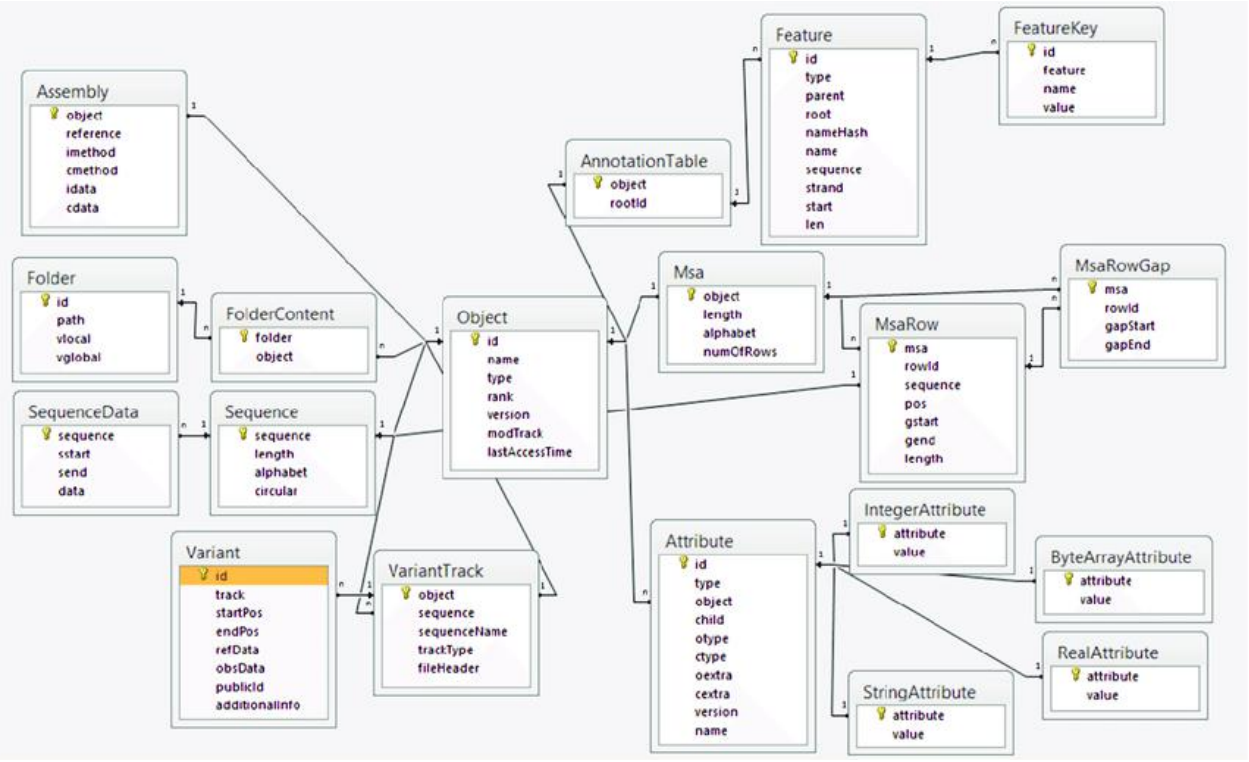

Fig 4.1 ER DIAGRAM

ER diagrams are used to model and design the relational databases, in terms of logic, business rules (in a logical data model) and in terms of the specific technology to be implemented (in a physical data model. In the above ER diagram the Object is the main Entity with Many-to-Many relationship, which has attributes like Id which is set as primary key attribute and name, type, rank, version, mod track, last access time is set as non key attribute . Object entity is linked with many other Entities like assembly, folder Content, Sequence, variant, variant track, attribute, Msa, Annotation Table, and Msaflow. Assembly table contains objects as primary key attribute and reference, method, method, idata, cdata are set as non key attribute, folder content table is connected with objects and Folder table so it is known as one-to-one relationship. In the folder content table Folder is set as foreign key attribute and object is set as non key attribute. In the folder table the id is set as primary key or foreign key, path, vlocal,

vglobal attribute are non key attributes and connected with folder content table. In the sequence table it is connected with the objects table and sequence data table so it is a one-to-one relationship. In a sequence entity the sequence is set as primary key and length, alphabet, circular attributes are set as non key attributes and it is linked with sequence data table, in that table sequence is set as foreign key attribute and start, send, data attributes are set as non key attribute. In annotation table entity the relationship is one-to-one relation cause it is connected with object table and feature table. In annotation table object is set as primary key attribute and root id attribute is set as non key attribute and it is connected to feature entity where id is set as primary key attribute and type, parent, root, nameHash, name, sequence, stand, start, len are set as non key attributes feature table is said to be one-to-one relationship cause it is in relation with annotation table entity and feature key entity. In feature key table the attribute id is set as primary key and feature, name, value attribute is set as non key attribute. Msa table is connected with msarow table and msarowgap table and object table so it is one-to-many relationship. In msa table the object is set as primary key attribute and length, alphabet, number of rows is set as non key attributes. In msarow entity it is connected to msa, object and macro gap so it is said as many-to-one relationship the attribute msa is set as primary key attribute and rowid, sequence, pos, gstart, gend, length is set as non key attribute. In msarowgap entity msa is set as primary key attribute and rowid, gap start, gap end is set as non key attribute. Attribute table is connected with object and integer attribute, byte array attribute, real attribute, string attribute tables so it is said to be a one-to-many relationship. In the attribute table the id is set as primary key attribute and type, object, child, otype, ctype, oextra, cextra, version, name is set as non key attribute. Integer attribute table contains attribute and value where attribute is set as primary key attribute and value is set as non key attribute. In byte array attribute table attribute is set as foreign key and value as non key attribute. In the StringAttribute table the attribute is set as foreign key attribute and value is set as non key attribute.

4.2 DATA DIRECTORY DIAGRAM

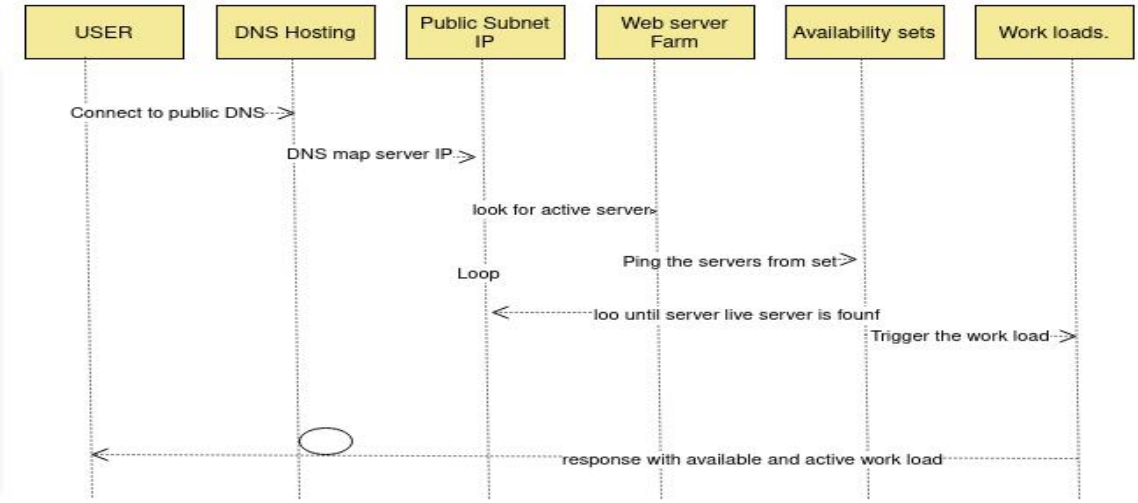

Fig 4.2 DATA DIRECTORY DIAGRAM

Data directory diagram is used to respond with active and available work load. In first the user is connecting to the public DNS Hosting and it lets the DNS map server internet protocol to public subnet IP, it looks for active server in Web Server Farm if no active server found it will search for active server again and again until it is found , if the active server is found then pings the server from availability set then it triggers the workload .

4.3 USER INTERFACE DIAGRAM (INPUT DESIGN)

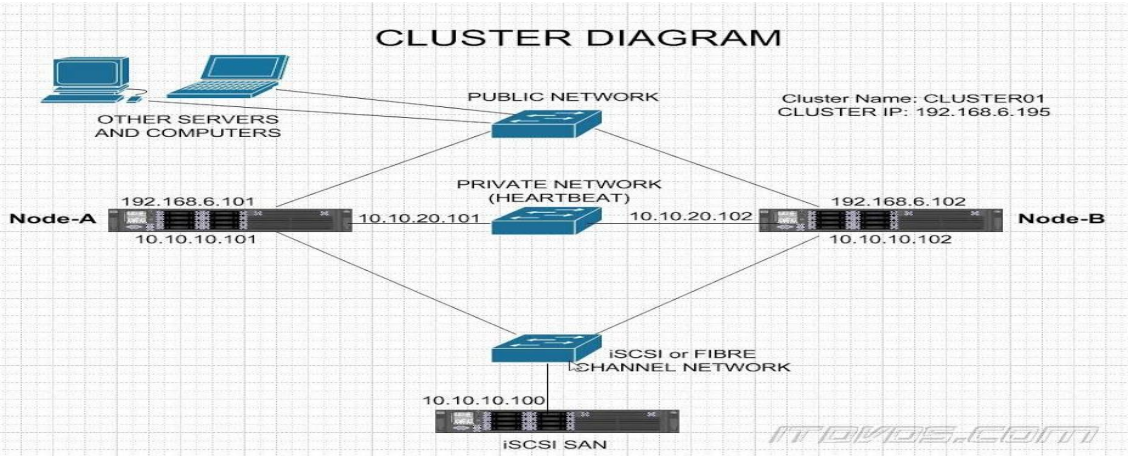

Fig 4.3 UI DIAGRAM

When multiple devices are requesting data it will enter into the public network and see which server is active. If the node A is an active server then the request gets into node A which has an ip address of 192.168.6.101 . Whenever any updation takes place then it will store into the active server and it replicates that updated data to the backup server so that whenever the active server crushes the backup server contains that data iscsi or fiber channel network is used to fetch data from the ISCSI SAN . iscsi san is a type of hardware device.

4.4 UML DIAGRAMS

4.4.1 USECASE DIAGRAM

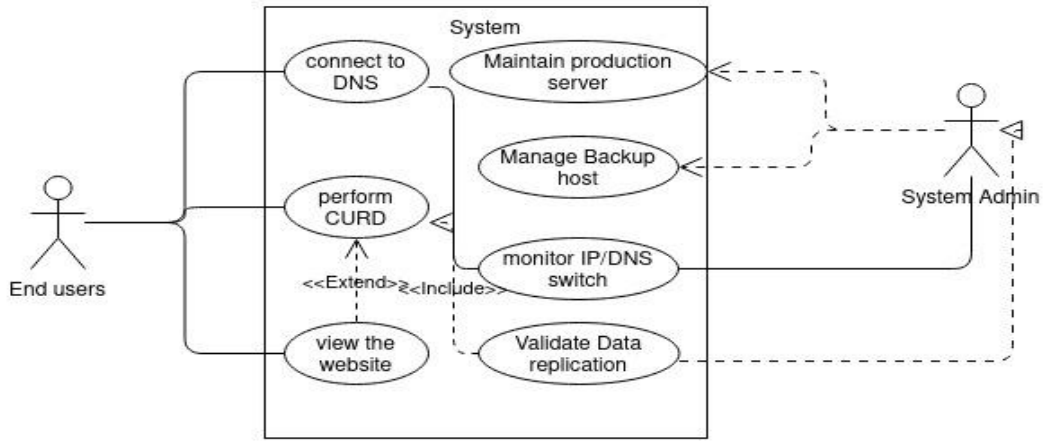

Fig 4.4.1 USECASE DIAGRAM

In use case diagram the end user is interacting with the system where he does any edit in system where the system is connected to DNS , the user can interact directly with the web server and can perform curd the system admin can monitor Ip and DNS switch and can validate data replication to which it occurs in another server system admin will maintain production server and manage backup host , monitoring IP/DNS switch and validate data replication can be done cause the system is connected with DNS ip address so any updation or edit is performed it is updated in server too.

4.4.2 CLASS DIAGRAM

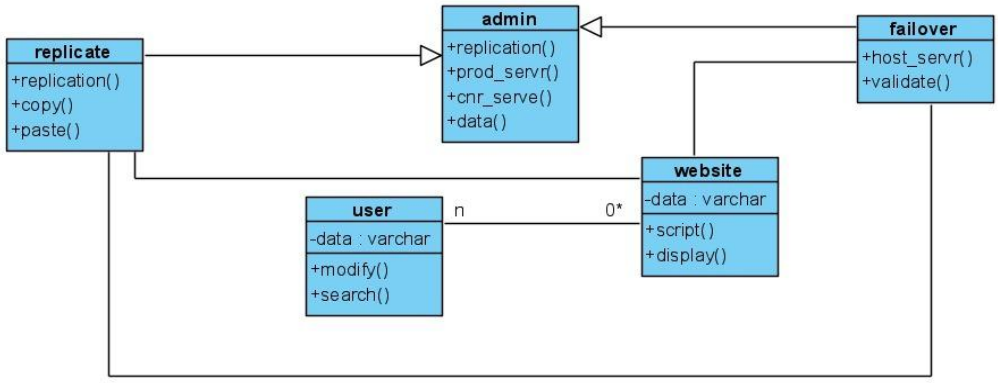

Fig 4.4.2 CLASS DIAGRAM

As we know that the class diagram is a static diagram, it represents the relation and the types of objects existing in the system. Here, in the class diagram of this system we have five classes. In the admin class the operations done here are replication, managing production and host server, then during failover the operations done are switching the IP to the backup so the delay can be reduced. During replication, the data is replicated to the backup for frequent intervals of time i.e., the user perform CRUD so they might add the data and hence attributes are to add data and the operations are to either to modify or search DNS website. In this class, a website is created to register the users.

CHAPTER 5
SYSTEM ARCHITECTURE

5.SYSTEM ARCHITECTURE

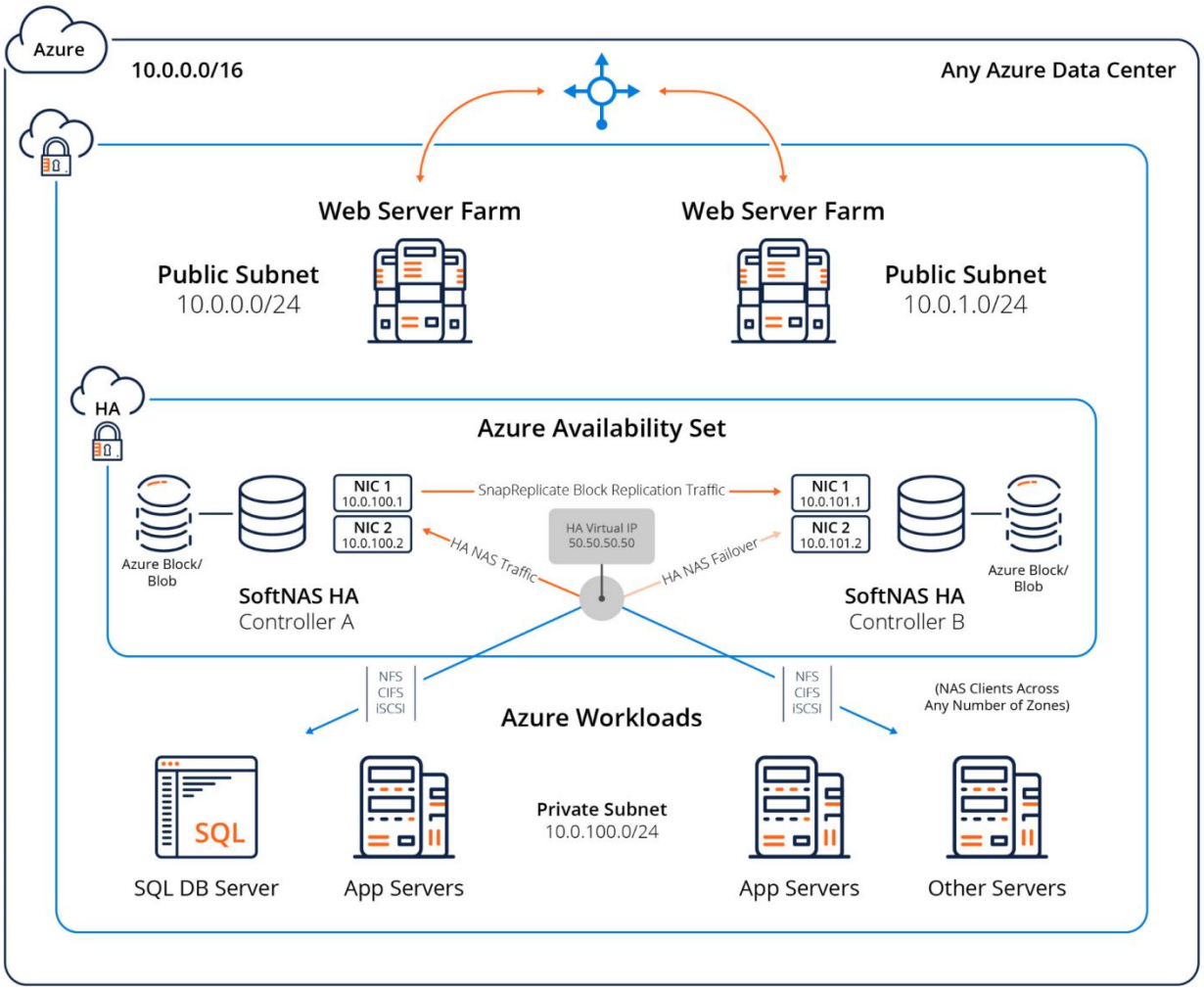

Fig 5 SYSTEM ARCHITECTURE

The figure illustrates the overview of disaster recovery mechanism with active storage replication. SoftNAS HA integrates seamlessly into today's virtualized data centres. In today's data centre, VMware is frequently used, with a network architecture consisting of several VLANs used to separate various types of network traffic. This architecture implements the best practice for SoftNAS HA. The DNS IP address in this model is 10.0.0/16, which is assigned to the Azure Data Centre. A switch is used to connect two web server farms of different servers with unique IP addresses. One server serves as the primary server, while the other, with IP address 10.0.1.0/24, serves as the backup server. Public user subnets are assigned outside the data centre network, and they must pass through one or more routers to get to the data centre. If server A crashes, the DNS name

is internally redirected to the backup server.

Azure Availability Set acts as storage unit with High Availability HA lock system, which acts as hardware architecture and are not interdependent. Here, SoftNAS block replication traffic is routed through a dedicated Replication VLAN and subnet, with SnapReplicate configured to flow across it. This prevents data replication traffic between controllers from interfering with storage or other data centre services. This is done using public Virtual IP address, which acts as an interface validating the data. Furthermore, all SNAP HA failover and takeover operations are controlled and witnessed by the HA Controller and also keeps a record of which storage controller is acting as the primary controller. Moreover, workloads are the programmes that run on the server. To virtualize the storage traffic, separate storage VLAN and subnets are assigned. Storage access is handled through NFS, CIFS and iSCSI file system between VMware vmKernels on each VM host.

5.1 MODULE DESIGN SPECIFICATION

There are 3 modules:

- Resource group creation
- Configure Test Web page
- Test and setup failover and replication

5.1.1 RESOURCE GROUP CREATION:

In the Azure portal, to form any resource, be it a VM, Disk, or Storage Account, the Azure Resource Group, is mandatory. Every Resource/Instance created are going be linked to a Resource Group. Thus, an Azure Active Directory group is created as they are Security Principals, which means they can be used to secure objects in Azure AD.

Process the DNS Name Server as it is the server that stores all DNS records for a domain,

which includes A records, MX records, CNAME records. Almost every domain rely on multiple name servers to increase reliability: if one name server goes down or if at all its unavailable then the DNS queries can go to another one.

Provision of PRODUCTION and BACKUP Server takes place. In this process a backup repository server is created in the cloud. This is where the protected data from your active source is stored.

5.1.2 CONFIGURE TEST WEBPAGE:

Scripting a Django based website to test the activeness of the server takes place here, as Django is a fully featured server-side web framework, which is written in pythonThis module shows you how to set up a particular development environment, and how to start using it to create your own specific web applications.

Domain name is purchased in GoDaddy as its used as a webhost and domain register and it supports websites and applications built using Django. However the company recommends choosing a VPS hosting plan or a dedicated server plan for the apps, once we choose a hosting plan, we can use the cPanel control panel to install python and Django, after the purchase of domain name we can host our website with the service.

Configure Apache knox for DNS failover, Knox provides connectivity-based failover functionality for service calls that can be made to more than one server instance in a cluster. Ha Provider configuration needs to be enabled for the service to enable this functionality and also the service itself must be configured with more than one URL in the topology file.

The Apache Knox Gateway is a system which is to extend the reach of Apache™ Hadoop® services to the users outside of a Hadoop cluster without reducing Hadoop Security, for users who access the cluster data and execute jobs knox simplifies the Hadoop security. And also the Knox Gateway is designed as a reverse proxy. knox offers load balancing by employing a simple round robin algorithm which prevents load on one

specific node.

5.1.3 TEST AND SETUP FAILOVER AND REPLICATION:

Host server using Nginx and Gunicorn. Nginx is a very powerful web server. A ton of things can be done using it, such as setting up reverse proxies or load balancing. It can also be used to host the static website, Gunicorn is used to pass request data to your application ,and to receive response data , it takes care of running multiple instances of web application, making sure they are healthy and restart them as needed , distributing incoming requests across those instances and communicate with the web server.
Multi-datacenter are designed to load balance the processing of data in multiple cluster, and also to safeguard against outages by replicating data across the clusters. If at all a particular datacenter goes down, the remaining datacenters have replicas of the data and can take over where the failed datacenter left off. Once the original datacenter recovers, it then resumes message processing. Replicator knows where to re-start data synchronization across clusters based on these supporting features for failover and disaster recovery. In case of failover from the production server to the CNR server, the consumers will start consuming data from the last committed offset.

5.2 ALGORITHM

5.2.1 Secure Dynamic Replication:

Secure dynamic replication (SDR) SDR is a heuristic and a security-based replication method that has three main steps as follows:

(1) Determination of popular file, The most needed files are selected for replication.
(2) Replica placement and optimize fuzzy rules with Competitive Swarm Optimizer

(CSO) .The popular file is divided into some fragments, and then, based on the T-coloring concept and CSO technique the suitable locations for storing fragments are determined.

(3) Replica replacement Due to the limited storage space, less valuable replicas are determined for deletion .We use SDR algorithm for securely replicating the file systems SDR algorithm enhances the security level of data without any encryption technique since each data center has only one part of the file, and as a result, even in a successful attack, useful information is not discovered. SDR strategy can handle the following cloud-specific attacks:

• Data Recovery It leads to a rollback of the virtual machine to a previous state and reveals the previous files.

• Cross-virtual machine attack It causes data breach by malicious virtual machine invading co-resident virtual machine.

• Improper media sanitization It leads to revealing data due to unworthy sanitization of storage elements.

• E-discovery Affords the disclosing data of one customer by confiscating hardware to investigate related to some other customers.

• Virtual machine escape, It leads to access to storage and computing elements by a malicious user or virtual machine escapes from the management of the virtual machine administrator.

ALGORITHM 1

1 Input: Number of individuals (L), Maximum number of rules (B), Number of data centers (N),T, q

2 Output:

Population P

Struct P{

```
        d,r,g,fit
 }
 Struct r{
        id,fuzzy_sets
}
for (i=1; i<=L; i++){
        P[i]. d= Build_merit_array (N); //Create merit array for i-th individual
        P[i]. r= Build_fuzzy_set_array (B, T); //Initial rules for i-th individual
        P[i]. g= Build_selection_array (B); //Create an array for selection rules that
are generated by i-th individual
        P[i]. fit= 0;
} //end for
```

- We use this algorithm to initialize the individuals ,where L is number of individuals and B is maximum number of rules and N is number of datacenters and open a for loop , for loop the program will execute from 1 to till the number of individuals found , and increment by 1. Inside the for loop

- P[i]. d= Build_merit_array (N); //Create merit array for i-th individual
  And P[i]. r= Build_fuzzy_set_array (B, T); //Initial rules for i-th individual

  And P[i]. g= Build_selection_array (B); //Create an array for selection rules that are generated by i-th individual and end the for loop

ALGORITHM - SDR

1. Input : Fp={F1,F2,.....,Fs} *//fp is list of popular files, DC={dc1,dc2,dc3.....,dcs}//set of data centers obtained by Best_individual from Algorithm 3*

2. OUTPUT: store fragments based on T-coloring
3. C={red_color , black_color};
4. DCs_path+ Find_short_path(DC) //return the shortest path of data centers to each other
5. For each Fk in set Fp{
6. Sore all data centers in DC into Set1;
7. Seth color of all Dcs to red_color;
8. k=1;
9. for (k=1;k<=F ; k++) //F is number of fragments{
10. Select the k-th data center (DCs) from Set1:
11. if(color of DCk = red_color){
12. data_centerX*=Distance(DCs_path,t); //*return all data centers at distance T from DCi and stores them in set Data_centerX**
13. Change color of all data centers in set Data_centerX* from red_color to black_color;
14. Store the select data center (Data_centerX) into Set2;
15. Remove DCk from Set1;
16. Insert DCk to set3;
17. }
18. }//*end for loop*

//Divide F1 into N fragments based on the merit of data centers in Set2

19. R_F= Replica(Fk) *//create a replica from selected popular file*
20. Divide R_F into F fragments;
21. Calculate Pn for each data centers in Set3;
22. }//*end for loop*

Fragmentation technique is used in this algorithm, this algorithm indicates the SDR

strategy, SDR strategy sorts all data centers according to the Merit in descending order, and it stores the sorting result into Set1 Then, it selects the N data centers from the top of Set1 by considering the color of them to store a fragment of replicas. SDR store N fragment of replica into N selected data center

The portion of the replica fragment is obtained as the following formula

$$P_x = \frac{T_x}{\sum_{j \in Set2} T_j}$$

where Tj denotes the merit value of data center j. We explain the proposed strategy with the following example based on N = 3. The list of popular files is F = {file6, file3, file5}, . For file6, it selects three data centers (Data center2, Data center1 and Data center4) from Set1 that is sorted based on Merit. Using Eq. the portion of file6 to be replicated on Data center2 is 50%. So we can replicate the first 50% of file6 on Data center2,the remaining 30% of file6 on Data center1 and the remaining 20% of file6 on Data Center4. It is interesting that even in successful violation, the SDR strategy guarantees that the hacker gets only one partition of the file. This is because keeping only one partition of a particular file in the data center indicates the flowchart of SDR strategy.

Time complexity of SDR algorithm

There are N data centers D={D1,..., DN} and M fles F={f1, …, fM}. We have three main steps (i.e., determination of popular fle, replica placement and replica replacement). For each step, the time complexity is explained as follows.

(1) Determination of popular fle This step has O(M log M) time, since M files must be

sorted based on access count in descending order.
(2) Replica placement Consider popular fles set Fp={f1,..., fS}, maximum generation K, L individuals and B maximum number of rules.

• Generation of the initial population for each popular fle takes O(LB + LN) time.
• The time complexity for finding high fitness value is O(N log N) (line 28 of Algorithm 10).
• Array d must be sorted for calculating fitness of each individual (line 29 of algorithm 10), so we have O(L × (N logN)).

5.2.2 NAS storage replication on mount

Since one of the main resources in the cloud is storage, tracking storage capacity is a key part of replication strategy. The percentage of storage usage is specified in below equation :

$$\frac{\textit{Storage usage = total\_storage space - available\_space}}{\textit{Total storage space}} \textit{ x 100}$$

Where Storage Usage is equal to the Total_Storage Space − Available_Space divided by Total_Storage Space × 100,Because data centers have limited storage space, replica replacement and replica placement are critical processes, especially when the resource cost is proportional to the amount that is being consumed. Different replication algorithms lead to different storage resource usage.

5.2.3 setup Ssh key and password less authentication

Azure currently supports the SSH protocol 2 (SSH-2) RSA public-private key pairs with a minimum length of 2048 bits. Other key formats such as ED25519 and ECDSA are not supported.

Create an SSH key pair:

Use the ssh-keygen command to generate both the SSH public and private key files. By default, these files are created in the ~/.ssh directory.To access the private key file you can specify a different location,and an optional password(passphrase) . If an SSH key pair with the same name which exists in the given location, those files are overwritten.

The following command creates an SSH key pair using RSA encryption and a bit length of 4096:

ssh-keygen -m PEM -t rsa -b 4096

Create your Virtual Machine with the az vm create command, you can optionally generate SSH public and private key files using the --generate-ssh-keys option. The key files are stored in the ~/.ssh directory unless they are specified otherwise with the --ssh-dest-key-path option. If an ssh key pair already exists and the --generate-ssh-keys option is used, a new key pair won't be generated but instead the prevailing key pair will be used. In the following command, replace VMname and RGname according to your own values:

az vm create --name VMname --resource-group RGname --image UbuntuLTS --generate-ssh-keys

Provide an SSH public key when deploying a VirtualMachine

To create a Linux VirtualMachine that uses SSH keys for authentication, specify your SSH public key when creating the VM using the Azure portal, Azure CLI, Azure Resource Manager. Display your public key with the subsequent cat command, replacing ***~/.ssh/id_rsa.pub*** with the path and filename of your own public key file if needed:

cat ~/.ssh/id_rsa.pub

A typical public key value looks like this example:

ssh-rsa
AAAAB3NzaC1yc2EAABADAQABAAACAQC1/KanayNr+Q7ogR5mKnGpKWRBQU7F
3Jjhn7utdf7Z2iUFykaYx+MInSnT3XdnBRS8KhC0IP8ptbngIaNOWd6zM8hB6UrcRTlTp
wk/SuGMw1Vb40xlEFphBkVEUgBolOoANIEXriAMvlDMZsgvnMFiQ12tD/u14cxy1WN
EMAftey/vX3Fgp2vEq4zHXEliY/sFZLJUJzcRUI0MOfHXAuCjg/qyqqbIuTDFyfg8k0JTty
GFEMQhbXKcuP2yGx1uw0ice62LRzr8w0mszftXyMik1PnshRXbmE2xgINYg5xo/ra3mq
2imwtOKJpfdtFoMiKhJmSNHBSkK7vFTeYgg0v2cQ2+vL38lcIFX4Oh+QCzvNF/AXoD
VlQtVtSqfQxRVG79Zqio5p12gHFktlfV7reCBvVIhyxc2LlYUkrq4DHzkxNY5c9OGSHXSl
e9YsO3F1J5ip18f6gPq4xFmo6dVoJodZm9N0YMKCkZ4k1qJDESsJBk2ujDPmQQeMjJ
X3FnDXYYB182ZCGQzXfzlPDC29cWVgDZEXNHuYrOLmJTmYtLZ4WkdUhLLlt5Xsdo
KWqlWpbegyYtGZgeZNRtOOdN6ybOPJqmYFd2qRtb4sYPniGJDOGhx4VodXAjT09om
hQJpE6wlZbRWDvKC55R2d/CSPHJscEiuudb+1SG2uA/oik/WQ==
username@domainname

copy and paste the contents of the public key file to use within the Azure portal or a Resource Manager template, ensure you don not copy any trailing whitespace. To copy a public key in macOS, you will be able to pipe the public key file to pbcopy. Similarly in Linux, you will be able to pipe the public key file to programs such as xclip. The public key that you just place on your Linux VM in Azure is by default stored in

~/.ssh/id_rsa.pub, unless you specified a special location when you created the key pair. With the public key deployed on your Azure VirtualMachine, and therefore the private key on your local system, SSH into your VM using the IP address or the DNS name of your VM. In the following command, replace azure user and myvm.westus.cloudapp.azure.com with the administrator user name and the fully qualified domain name (or IP address):

ssh azureuser@<ipaddress>

If you're connecting to this VM for the first time, you'll be asked to verify the host's fingerprint. It is tempting to easily accept the fingerprint that's presented, but that approach exposes you to a possible person-in-the-middle attack. You must always validate the host's fingerprint. You should do this only the first time you connect from a client. To get the host fingerprint via the portal, use the Run Command feature to execute the command ssh-keygen -lf ***/etc/ssh/ssh_host_ecdsa_key.pub | awk '{print $2}'.***

5.2.4 Setup apache Knox for dns failover

Apache Knox Gateway (Apache Knox) is the Web/REST API Gateway solution for Hadoop and it provides a single access point for all of the Hadoop resources over REST. The Knox Gateway also enables the integration of enterprise identity management solutions and various perimeter security features for REST/HTTP access to Hadoop.

Knox is installed on kerberized and non-kerberized clusters. The following instructions are used to complete the installation process of knox:

Install the Knox Package on the Knox Server

To install the Knox RPM or package, run the following command as root:

RHEL/CentOS/Oracle Linux:

sudo yum install knox

For Ubuntu:

apt-get install knox

The installation creates the following:

knox user in /etc/passwd

Knox installation directory: /usr/hdp/current/knox-server, referred to as $gateway_home

Knox configuration directory: /etc/knox/conf

Knox log directory: /var/log/knox

Set up and Validate the Knox Gateway Installation

This section explains us how to get the gateway up and running, and also how to test access to our existing cluster with the minimal configuration.

To set up the gateway and test access:

Set the master secret.

su -l knox -c "$gateway_home/bin/gateway.sh setup"

You are prompted for the master secret. Now Enter the password at the prompt.

Start the gateway:

su -l knox -c "/usr/hdp/current/knox-server/bin/gateway.sh start"

Starting Gateway succeeded with PID 1871.

The gateway starts. The PID is stored in /var/run/knox.

Start the demo LDAP service that contains The guest user account for testing.

```
su -l knox -c "/usr/hdp/current/knox-server/bin/ldap.sh start"
Starting LDAP succeeded with PID 1965.
```

In a production environment, use AD or OpenLDAP for authentication. For detailed instructions on configuring of the Knox Gateway, see Configuring Authentication that is given in the Hadoop Security Guide.

Verify that the gateway and LDAP service are running:

```
su -l knox -c "$gateway_home/bin/gateway.sh status"
Gateway is running with PID 1871.
su -l knox -c "$gateway_home/bin/ldap.sh status"
LDAP is running with PID 1965.
```

Confirm access from the gateway host to the WebHDFS Service host using telnet:

To enable telnet set dfs.webhdfs.enabled to true.

```
telnet $webhdfs_host $webhdfs_port
```

You must be able to reach the internal cluster service from the machine on which Knox is running before continuing.

Update the Web HDFS:

The WebHDFS host information is located in the $gateway_home/conf/topologies/sandbox.xml file.

Find the service definition for WebHDFS and update it as follows:

<service>
<role>WEBHDFS</role>
<url>http://$webhdfs_host:$webhdfs_port/webhdfs</url>
</service>

where $webhdfs_host and $webhdfs_port (default port is 50070) match your environment.

On an external client that has curl, enter the given command :

Curl -k -u guest:guest-password -X GET
"https://$gateway_host:8443/gateway/sandbox/webhdfs/v1/?op=LISTSTATUS"

where sandbox is the name of the cluster topology descriptor file that is which you created for testing. If you renamed it, then replace sandbox in the above command.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.SYSTEM IMPLEMENTATION

6.1 Server Setup and provisioning(Azure)

Expedite Linux VM Machine creation:

```
#cloud-config
package_upgrade: true
packages:
  - nginx
write_files:
  - owner: www-data:www-data
        path: /etc/nginx/sites-available/default
        content: |
        server {
        listen 80 default_server;
        server_name _;
        location / {
        # First, try if the file exists locally, otherwise request it from the app
        try_files $uri @app;
        }
        location @app {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
```

```
        }}
    runcmd:
      # install Node.js
      - 'curl -sL https://deb.nodesource.com/setup_16.x | sudo -E bash -'
      - 'sudo apt-get install -y nodejs'
      # clone GitHub Repo into myapp directory
      - 'cd /home/azureuser'
      - git clone "https://github.com/Azure-Samples/js-e2e-vm" myapp
      # Start app
      - 'cd myapp && npm install && npm start'
      # restart NGINX
      - systemctl restart nginx
```

CREATE A VM RESOURCE:

```
az vm create \
--resource-group DNS-project-rg \
--name DNSProd-vm \
--location eastus \
--public-ip-sku Standard \
--image UbuntuLTS \
--admin-username azureuser \
--generate-ssh-keys \
--custom-data cloud-init-github.txt
```

 # for CNR Server

```
az vm create \
--resource-group DNS-project-rg \
```

```
--name DNSCnr-vm \
--location eastus \
--public-ip-sku Standard \
--image UbuntuLTS \
--admin-username azureuser \
--generate-ssh-keys \
--custom-data cloud-init-github.txt
```

OPEN PORT FOR AZ VM:

```
az vm open-port \
--port 80 \
--resource-group rg-demo-vm-eastus \
--name demo-vm
```

```
const os = require('os');
const express = require('express')
const app = express()

app.use('/public', express.static('public'))
app.get('/', function (req, res) {

        const clientIP = req.headers['x-forwarded-for'];
        const msg = `HostName: ${os.hostname()}<br>ClientIP:
${clientIP}<br>DateTime: ${new Date()}<br><img width='200' height='200'
src='/public/leaves.jpg' alt='flowers'>`
        console.log(msg)

        res.send(msg)
```

```
})
app.listen(3000, function () {
        console.log(`Hello world app listening on port 3000! ${Date.now()}`)
})
```

6.2 Create Django site to test in local

REQUIREMENTS.TXT:

```
asgiref==3.4.1
Django==3.2.8
psycopg2-binary==2.9.1
pytz==2021.3
sqlparse==0.4.2
```

DRIVER CODE:

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
        """Run administrative tasks."""
        os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'dnsfailoverproj.settings')
        try:
        from django.core.management import execute_from_command_line
        except ImportError as exc:
```

```
        raise ImportError(
        "Couldn't import Django. Are you sure it's installed and "
        "available on your PYTHONPATH environment variable? Did you "
        "forget to activate a virtual environment?"
        ) from exc
        execute_from_command_line(sys.argv)

if __name__ == '__main__':
        main()
```

BASE TEMPLATE JINJA FILE:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta http-equiv="content-type" content="text/html;charset=utf-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-
to-fit=no">
        <meta name="description" content=" KAVWV " />
        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3Ahi
U" crossorigin="anonymous">
        <link rel="stylesheet" href="{% static 'css/base.css' %}">
        <!--{% block link %}   {% endblock %}-->
```

```
        <title> DNS Failover Test site</title>
</head>
<body class="scrole_bg">
        <nav class="navbar shadow navbar-expand-lg navbar-dark " id="navb">
        <!-- Navbar content ---->
        <a class="navbar-brand " href="{% url 'home' %}">
        <span style="letter-spacing: 2pt; word-spacing: 1pt; font-weight: bold;">
                Testing DNS Failover
        </span>
        </a>
        <a class="navbar-brand " href="{% url 'register' %}">
        <span style="letter-spacing: 2pt; word-spacing: 1pt; font-weight: bold;">
                registration
        </span>
        </a>
        </nav>
<!---------------------- Body Block ------------------------------>

{% block content %} {% endblock %}

<!--------------------- Footer ------------------------------------>
        <!-- Footer -->
        {% block aditionaljs %} {% endblock %}



</body>
```

JINJA FILE TO TEST REPLICATION:

```
{% extends 'base.html' %}
{% load static %}

{% block content %}
<div style="color: black;margin-top: 5%;padding: 1%;">
        <h1>hello! please Register to test the storage Replication. </h1>

        <form action="{% url 'home' %}" method="POST">
        {% csrf_token %}
        <div class="form-group">
        <label for="name">name</label>
        <input type="text" class="form-control" id="name" name="name"
placeholder="name" required>
        </div>
        <div class="form-group">
        <label for="email">Email address</label>
        <input type="email" class="form-control" id="email" name="email" aria-
describedby="emailHelp" placeholder="Enter email" required>
        <small id="emailHelp" class="form-text text-muted">We'll never share your
email with anyone else.</small>
        </div>
        <div class="form-group">
        <label for="college">college</label>
        <input type="text" class="form-control" id="college" name="college"
placeholder="college" required>
        </div>
        <div class="form-group">
        <label for="contact">contact</label>
```

```
        <input type="text" class="form-control" id="no" name="no"
placeholder="contact" required>
        </div>
        <div class="form-group">
        <label for="location">location</label>
        <input type="text" class="form-control" id="loc" name="loc"
placeholder="location" required>
        </div>
        <button type="submit" class="btn btn-primary" >Submit</button>
        </form>

</div>
{% endblock %}
{% block aditionaljs %}
{% endblock %}
```

JINJA FILE TO TEST FAILOVER:

```
{% extends 'base.html' %}
{% load static %}

{% block content %}
<div style="color: black;margin-top: 5%;padding: 1%;">
        <h4 style="padding: 1%; color: red;"> {{ data }} </h4>
        <img src="{% static 'images/images.png' %}" alt=" image here "
        height="100px" width="100px" style="border-radius: 50%;float: right;">

        <h2>Welcome to the DNS Failover Testing page. <br>Connected Server details:
<br></h2> <h3 style="color: darkblue;"><b>{{ regdata }}</b> </h3>
```

```
        <ul>
        <li >
        The above host name and host IP address will change based on the Active server
        </li>
        <li >
        For Your information please find the IP address of our Production and Backup server
        <br>
                    Production Server: 104.211.9.209 (10.0.0.4) Name : Dnsprodvm<br>
                    Backup server: 20.124.99.148 (10.0.0.5) Name: DnsCNRvm

        </li>
        </ul>

</div>
{% endblock %}
```

PYTHON SCRIPT TO VALIDATE:

```
from django.shortcuts import render
from django.http import HttpResponse, request, HttpResponseRedirect
from monthlych.models import Register
import socket
import subprocess


# Create your views here.
```

```
def home(request):
        hostname=socket.gethostname()
        hip = subprocess.run(['curl','ifconfig.me'],stdout=subprocess.PIPE)
        regdata = 'Host name : {hostname} \n Host Ip :{hostip}
'.format(hostname=hostname,hostip='{a}
( {b} )'.format(a=hip.stdout,b=socket.gethostbyname(hostname)))
        if request.method =='POST':
        f_data=request.POST
        name= f_data['name']
        email=f_data['email']
        college= f_data['college']
        contact = f_data['no']
        loc = f_data['loc']
        #hresp = "<p>"+name+ email +college+contact+loc+ "<p>"
        reg = Register.objects.create(name= name,mailid=email,college=
college,contact=contact,loc=loc)
        #return HttpResponse(hresp)
        return render(request,'monthlych/index.html',{'regdata':regdata,'data':'you have
registered your information, please check the admin portal'})
        #return HttpResponse("<h1>iam inside chal</h1>")
        elif request.method =='GET':

        return render(request,'monthlych/index.html',{'regdata':regdata})

def regis(request):
        return render(request,'monthlych/register.html')
```

SETUP MODEL VIEW (MVC):

```
from django.db import models

# Create your models here.
class Register(models.Model):
        name = models.CharField(max_length=100)
        mailid = models.EmailField(max_length=100)
        college = models.TextField()
        contact = models.BigIntegerField()
        loc = models.CharField(max_length=20)

        def __str__(self):
        return '{0} from {1}'.format(self.name,self.college)
```

GLOBAL PARAMETERS:

```
from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent



# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/
# Application definition
INSTALLED_APPS = [
        'monthlych.apps.MonthlychConfig',
        'django.contrib.admin',
```

```
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
]
MIDDLEWARE = [
        'django.middleware.security.SecurityMiddleware',
        'django.contrib.sessions.middleware.SessionMiddleware',
        'django.middleware.common.CommonMiddleware',
        'django.middleware.csrf.CsrfViewMiddleware',
        'django.contrib.auth.middleware.AuthenticationMiddleware',
        'django.contrib.messages.middleware.MessageMiddleware',
        'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'dnsfailoverproj.urls'
 TEMPLATES = [
        {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
        'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
        ],
```

```
        },
        },
]

WSGI_APPLICATION = 'dnsfailoverproj.wsgi.application'
'''DATABASES = {
        'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
        }
}'''


# Password validation
# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators
AUTH_PASSWORD_VALIDATORS = [
        {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
        },
        {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
        },
        {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
        },
        {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
        },
```

```
]

# Internationalization
# https://docs.djangoproject.com/en/4.0/topics/i18n/

LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC_ROOT=os.path.join(BASE_DIR,'static')
STATIC_URL = '/static/'
STATICFILES_DIRS = [
        os.path.join(BASE_DIR,'dnsfailoverproj/static')
]

# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
try:
        from .local_settings import *
except ImportError:
        pass
```

6.3 Configure DNS and Load balancer

```
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'djangfjfsdufhwejko-insecure-
pkw&=zsas(ffnaafff3s0g=x=h@i_k_52wewewe(mje5v17d%x!5#vzgf(a28glmkk'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = [‘ip of host 1’, ‘ip of host 2’, DNS]
DATABASES = {
        'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': database name
        'USER': postgres
        'PASSWORD':password
        'HOST': 'finalyearpostgresql.postgres.database.azure.com',         }
        }
```

6.4 Deploy Web app with Nginx and Gunicorn

```
-- install python and other necessary file
#sudo apt install python3-pip python3-dev libpq-dev postgresql postgresql-contrib nginx
curl python3-venv

-- create a requirement.txt and push it github $pip freeze > requirements.txt
-- connect to postgresql and create a database, user and grant certain previliges
-- push the code to a github repository
```

-- navigate to server and create a dir named pyapp and create a venv inside it.
-- clone the git repo in pyapp and source the venv
-- install requirements.txt inside venv $pip install -r requirements.txt
-- add local_settings.py in the server
--# Run Migrations
```------python manage.py makemigrations
```------python manage.py migrate
-- create superuser in django and set password -- python manage.py createsuperuser
-- Create static files -- python manage.py collectstatic
-- create exception for port 8000 in firewall $sudo ufw allow 8000 and set networking property to allow 8000 in azure
-- put public ip in local_settings.py and run the server, (static files will not be applied at this stage(as debug =false so nginx will take care of static))
-- setup gunicorn to serve in port 80 and make it prod ready
-- install gunicorn using pip $pip install gunicorn and run $gunicorn --bind 0.0.0.0:8000 kavwv.wsgi
-- stop and deactivate the venv in the server

GUNICORN SETUP

-- open gunicorn socet file and past the folowing lines:
$ sudo nano /etc/systemd/system/gunicorn.socket
```
[Unit]
Description=gunicorn socket

[Socket]
ListenStream=/run/gunicorn.sock
```

```
[Install]
WantedBy=sockets.target
```

-- Open gunicorn.service file and past the folowing lines:

```
$ sudo nano /etc/systemd/system/gunicorn.service

[Unit]
Description=gunicorn daemon
Requires=gunicorn.socket
After=network.target

[Service]
User=dbuser-az
Group=www-data
WorkingDirectory=/home/dbuser-az/pyapps/kavwv-django
ExecStart=/home/dbuser-az/pyapps/venv/bin/gunicorn \
        --access-logfile - \
        --workers 3 \
        --bind unix:/run/gunicorn.sock \
        kavwv.wsgi:application

[Install]
WantedBy=multi-user.target
```

-- start and enable gunicorn socket

```
# sudo systemctl start gunicorn.socket
# sudo systemctl enable gunicorn.socket
```

-- check status of gunicorn $sudo systemctl status gunicorn.socket

-- check existance of gunicorn socket $file /run/gunicorn.sock

SETUP NGINX :

-- create project folder

$sudo nano /etc/nginx/sites-available/kavwv-django

--add the content for the file to listen to port 80

```
server {
        listen 80;
        server_name YOUR_IP_ADDRESS;

        location = /favicon.ico { access_log off; log_not_found off; }
        location /static/ {
        root /home/dbuser-az/pyapps/kavwv-django;
        }

        location /media/ {
        root /home/dbuser-az/pyapps/kavwv-django;
        }

        location / {
        include proxy_params;
        proxy_pass http://unix:/run/gunicorn.sock;
        }
}
```

-- Enable the file by linking to the sites-enabled dir

$sudo ln -s /etc/nginx/sites-available/kavwv-django /etc/nginx/sites-enabled

-- test nginx config $sudo nginx -t

-- restart nginx $ sudo systemctl restart nginx

-- delete firewall rule 800 which is alowed $sudo ufw delete allow 8000

-- open up firewall to allow traffic to port 80 where nginx runs $sudo ufw allow 'Nginx Full'

You will probably need to up the max upload size to be able to create listings with images

Open up the nginx conf file $sudo nano /etc/nginx/nginx.conf

sudo systemctl restart nginx

sudo systemctl restart gunicorn

CHAPTER 7

PERFORMANCE ANALYSIS

7.PERFORMANCE ANALYSIS

7.1 LOAD BALANCING

Load balancing refers to a set of approaches for distributing workload and traffic among multiple network servers.In layman's words, the premise is straightforward: the more hands on deck, the less labour each person has to perform and the more effectively a job may be completed.This " community labour" in a computer network environment aids overall computing efficiency by increasing throughput, optimizing performance, and minimizing downtime.

Local vs. Global load balancing

Originally, load balancing meant distributing traffic across servers in a single location-say,a single data center,Today's computer,on the other hand,is increasingly internet and worldwide.As a result, load balancing has gained considerably larger meaning.

Traditional load balancing principles Global Server Load Balancing (GSLB).Workload distribution,on the other hand,is no longer limited to a single local network or data center. Rather, work is distributed globally (cross data center). This poses a number of additional issues for modern load balancing solutions, as they must consider critical communications aspects such connection quality across geographically scattered facilities at any given time, as well as the actual requester's geographical location.

As described below, early generation GSLB solutions relied on DNS Load Balancing, which limited their performance and efficiency. Cloud computing now offers an excellent solution for both local and global load balancing, allowing a single cloud-based service to manage both scenarios.

DNS: Un-synchronized and not load-aware

DNS solutions,which were formerly the gold standard for global server load balancing, are now only acceptable for small apps or websites. DNS Load Balancing is increasingly being recognized as unable meet the standards of enterprise-class GSLB.

To begin, DNS-based systems employ load balancing pools are tailored to specific are geographic regions. The administrator of the load balancer determines which web servers are available and how often traffic should be directed to them. This allows for maximum utilization of geographically distributed infrastructure in theory.DNS load balancing,on the other hand,uses a round robin distribution approach. The load balancer goes through a list of servers and sends one connection to each one in turn.when it hits the end of the list, it restarts from the beginning.The issue is that the round robin distribution is not load aware,which means the load balancer has no way of knowing whether the next server in line is indeed the best option. As a result,the server is either underutilised or overburdened.Furthermore, DNS records do not have built-in failure detection.This means that even requests if a server is not responding,queries can be forwarded to the next server on the list.Finally, DNS-based load balancing can cause delays. This is because a DNS change must be reported at the ISP level in order to take effect.Every TTL (Time to Live) cycle, ISPs only refresh their DNS cache once. This implies that ISPs will be uninformed of the change until the cache is refreshed, and traffic will continue to be routed to the incorrect server. Workarounds for this issue have been created, such as setting TTL to a low value,however these might have a detrimental impact on performance and still not ideal.

By way of example, consider the following scenario:

Server A and Server B are the two servers that run a certain application.The TTL cycle is one hour long (a common value).

The load on Server A can increase at any time,causing the DNS load balancer needs to start routing traffic to Server B.

what if the traffic surge occurs when the TTL cycle is still 20 minutes away from being updated? The redistributed of the load would take 20 minutes in the scenario.

Meanwhile, all traffic would continue to be directed to the overburndened server, resulting in increasing delays, slower delivery,and service degradation—even service failure. Some ISPs will keep a DNS cache after 20 minutes. This means that a variable percentage of traffic will still be routed incorrectly, resulting in inconsistent performance.

Environment parameter	Virtual Platform	Service Cluster Occupation	Real Servers Occupation
CPU	2.27 GHz *4	45MHz * 15	2.27G * 6
Memory	144 G	620M * 15	4 G * 6
Disk	20 T	8G * 15	70G * 6

Table 7.6 Performance analysis

When compared to the service provided by Real Servers, the results show that the Service Cluster reduces DNS service failover time by 94% (1.53 vs. 26.38)

7.2 CONFIGURATION PARAMETER:

Test	DNS Frontend	Number clients	clients
1-8	1 recursor	1	type: wired sendRate:{100,250,500,1000} records:{A,NAPTR}
9-16	1 recursor	3	type: all sendRate:{100,250,500,1000} records:{A,NAPTR}
17-20	1 recursor	5	type: all sendRate:{100,250,500,1000} records:{A,NAPTR}
21-24	1 dnsdist least-Outstanding	5	type: all sendRate:{500,1000} records:{A,NAPTR}
25-28	1 dnsdist firstAvailable	5	type:all sendRate:{500,1000} records:{A,NAPTR}
29-36	1 recursor	1	type: wireless sendRate:{100,250,500,1000} records:{A,NAPTR}
37-40	1 recursor	25	type: all sendRate:{500,1000} records:{A,NAPTR}
41-44	1 recursor	25	type: all sendRate:{500,1000} records:{A,NAPTR}
45-48	1 dnsdist least-Outstanding	25	type: all sendRate:{500,1000} records:{A,NAPTR}

Table 7.7 Configuration Parameters

According to the aforementioned analysis, the system has been tested in a harsh The above analysis shows that the system is tested in a toughened environment with all corner cases and is capable of 94% uptime.

R1	R2	R3	Response time
0.2	0.4	0.4	72.00
0.4	0.4	0.4	74.50
0.4	0.2	0.4	72.90
0.5	0.1	0.4	72.80
0.4	0.1	0.5	75.09
0.4	0.3	0.4	74.33

Table 7.7.1 Response time for Replication(in seconds)

Furthermore, the data replication problem must take into account security levels, work priority, and other criteria so that service providers may meet the majority of service-level agreements by enhancing system usage.

CHAPTER 8
CONCLUSION

8.CONCLUSION

8.1 Results and discussion:

An elastic and scalable architecture for DNS as a Service, suitable for cloud-based platforms, was presented and validated. The provided assessment, obtained through experimentation, demonstrated the employed scaling practices for cloud-based services. The obtained results further proved the proposed architecture's elasticity and flexibility, which takes into account the specificities of the DNS service while not being tied to a particular cloud-computing platform. Regarding the performance evaluation of the presented architecture, it became clear that the service is able to accommodate variable loads of DNS queries per second, always keeping satisfactory levels of performance in terms of DNS queries throughput and low latency DNS answers. This performance level was maintained by DNSaaS resorting to a horizontal scaling approach, instantiating additional resources whenever required. The DNS Service Cluster has a greater number of SQRs and reduces failover time by 94% compared to the Real Servers.

8.2 Future Enhancements:

The future scope of DNS and Store failover and replication involves addressing the following key areas:

1. Increase Replication rate in Cluster based environment.
2. Overcome lag of 6% in DNS failover which can be high in fast paced cluster regions.

APPENDICES

A.1 SAMPLE SCREENS

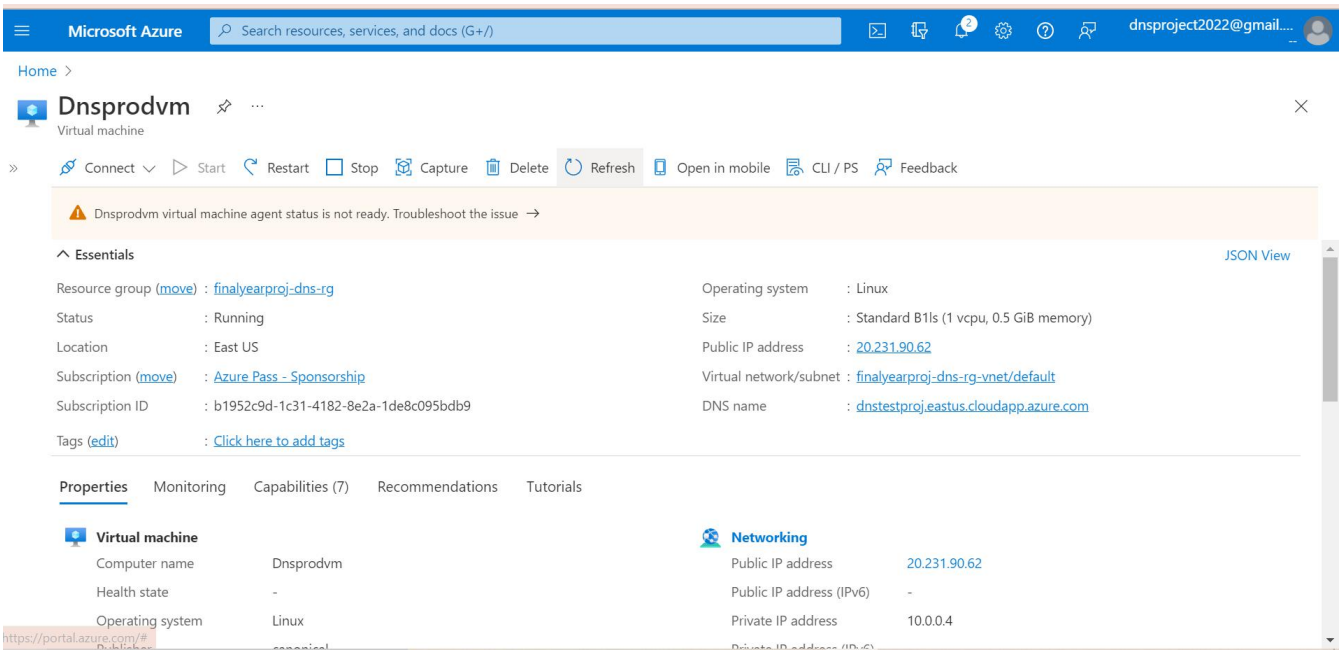

Fig A.1.1 DNS Production Server VM

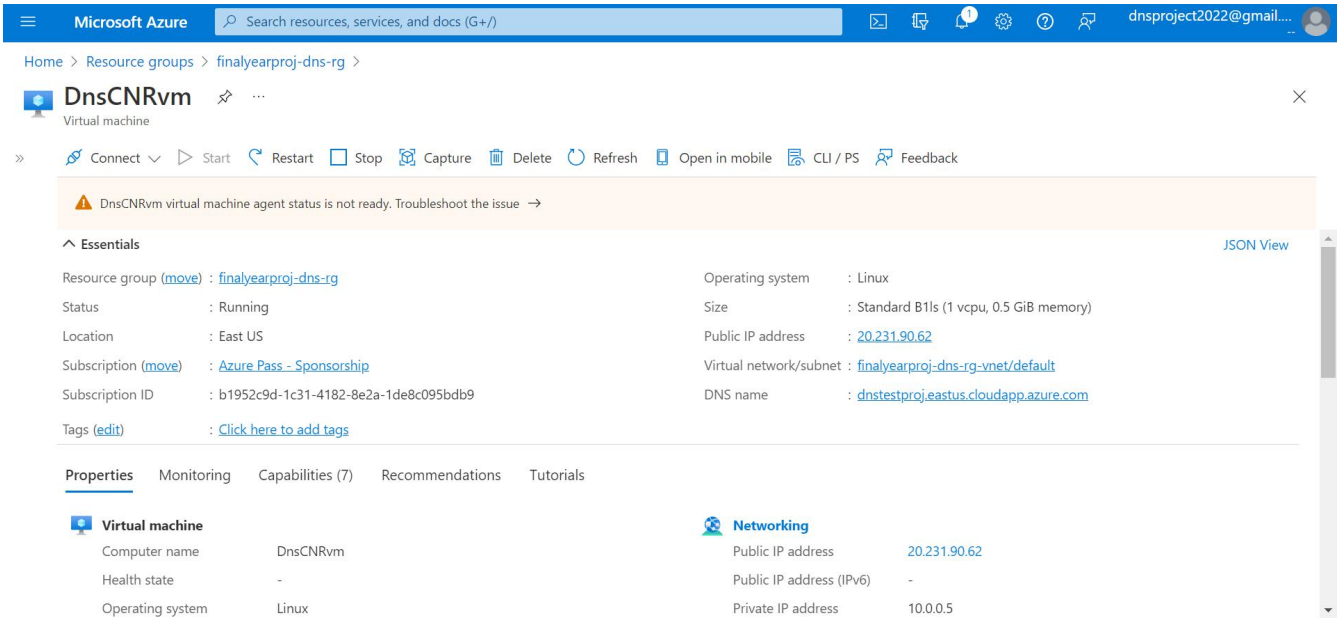

Fig A.1.2 DNS Contingency Server VM

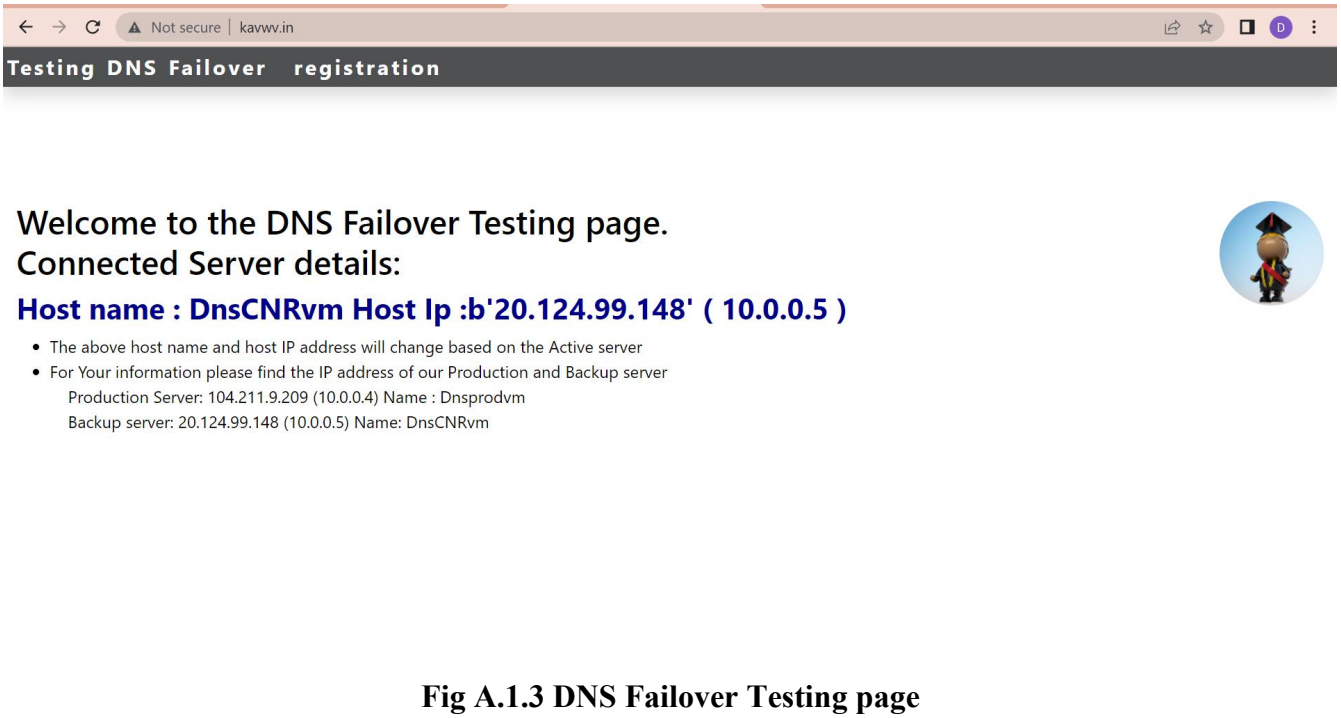

Fig A.1.3 DNS Failover Testing page

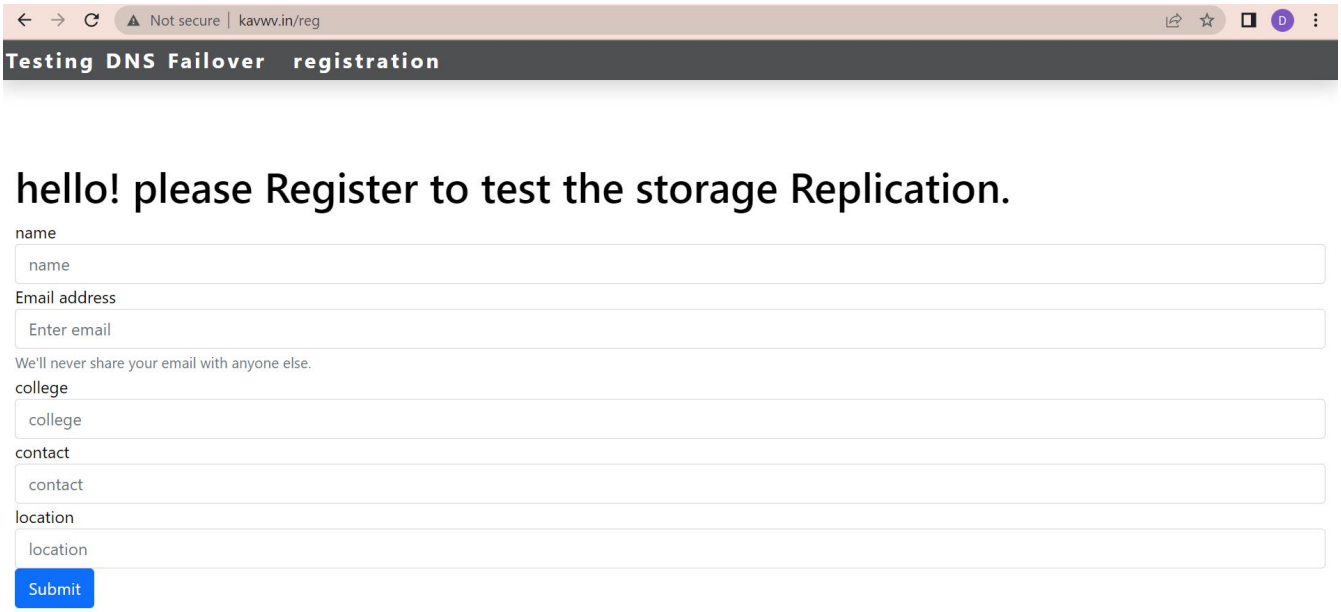

Fig A.1.4 DNS registration page

Fig A.1.5 Production Server login page

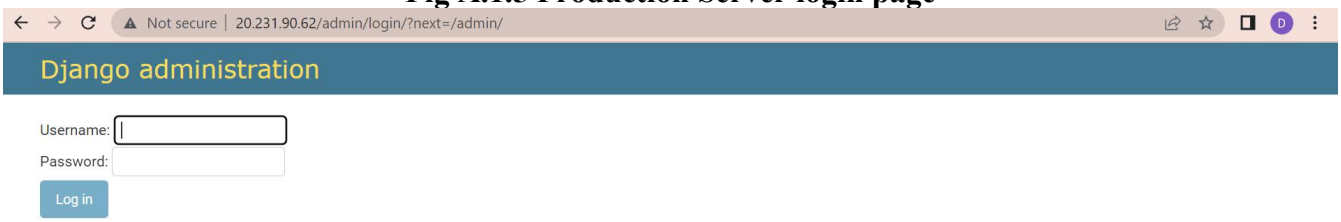

Fig A.1.6 Contingency Server admin page

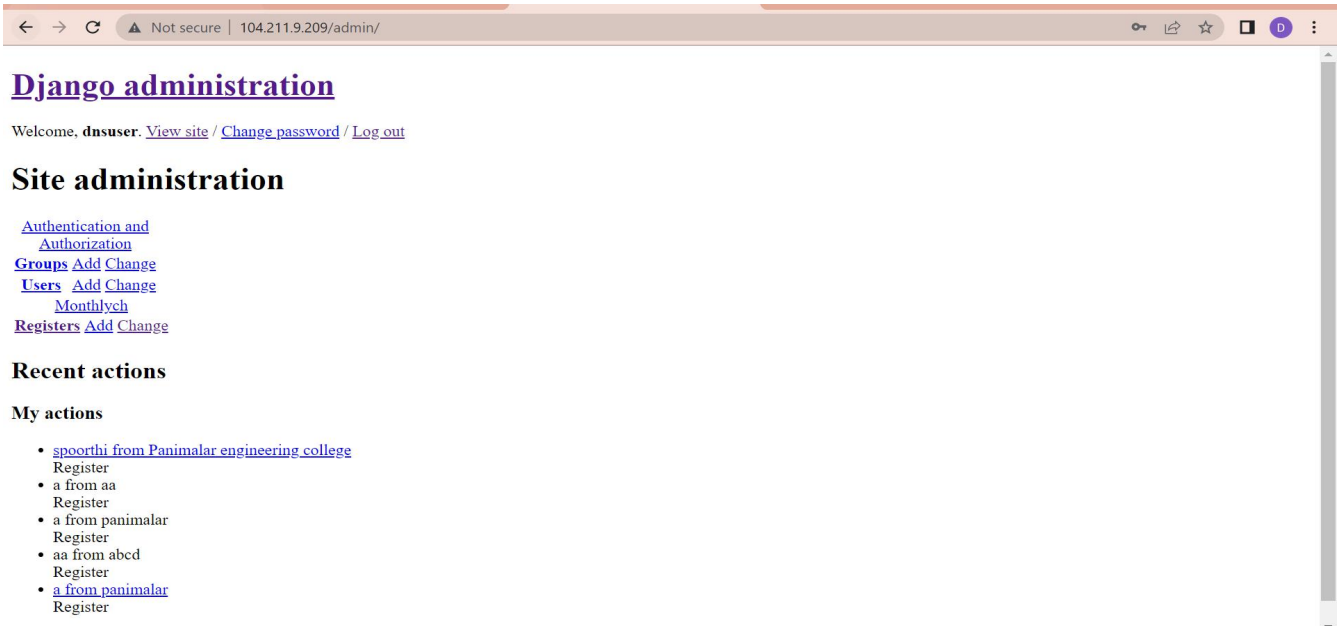

Fig A.1.7 Django Adminstration page

REFERENCES

[1] S. N. John and T. T. Mirnalinee, "A novel dynamic data replication strategy to improve access efficiency of cloud storage," Information Systems and e-Business Management , vol. 18, pp. 405-426, July. 2019.

[2] N. Mansouri, M. M. Javidi and B. M. H. Zade, "A CSO-based approach for secure data replication in cloud computing environment," The Journal of Supercomputing, vol 77, pp. 5882-5933, 2021.

[3] N. Mansouri, M. M. Javidi and B. M. H. Zade, "Hierarchical data replication strategy to improve performance in cloud computing," Frontiers of Computer Science, vol 15, Article 152501, 2021.

[4] Yanling Shao, Chunlin Li, Zhao Fu, Jia Leyue and Luo Youlong, "Cost-effective replication management and scheduling in edge computing," Journal of Network and Computer Applications, vol 129, pp. 46-61, March. 2019.

[5] Xiong Fu, Jian Li, Wenjie Liu, Song Deng and Junchang Wang, "Data replica placement policy based on load balance in cloud storage system," IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC 2019).

[6] A. U. Rehman, R. L. Aguiar and J. P. Barraca, " Fault-Tolerance in the Scope of Software-Defined Networking (SDN)," IEEE Access, Vol 7., to be published.

[7] S. Zheng, M. Hoseinzadeh, and S. Swanson, "Ziggurat: A tiered file system for non-volatile main memories and disks," in *Proc. FAST'19,* Boston, MA, USA, February 25–28, 2019.

[8] V. Bindhu, "Constraints Mitigation in Cognitive Radio Networks Using Cloud Computing," Journal of trends in Computer Science and Smart technology, vol 2,pp. 1-14, 2020.

[9] Ernesto Garbarino, "Beginning Kubernetes on the Google Cloud Platform: A Guide to Automating Application Deployment, Scaling, and Management," pp. 129-154, 2019.

[10] Jayashree Mohan, Amar Phanishayee, Vijay Chidambaram, “CheckFreq: Frequent, Fine-Grained DNN Checkpointing,” in *Proc.* 19th USENIX Conference on File and Storage Technologies, February 23–25, 2021.

[11] L. Y. Chen, W.M. Li, and Z. M. Lei, “Alleviating the Impact of DNS DDoS Attacks,” Networks Security, Wireless Communications and Trusted Computing, International Conference, Apr. 2010, pp. 240- 243.

[12] A. J. Kalafut, C. A. Cole, C. Lei, M. Gupta, and N. E. Myers, “An empirical study of orphan DNS servers in the internet,” In IMC '10: Proceedings of the 10th annual conference on Internet measurement, Melbourne, Nov. 2010, pp. 308-314.

[13] P. Vixie, “DNS Complexity,” ACM Queue vol. 5, no. 3,Apr. 2007.

[14] D. Wessels, "A Recent DNS Survey," DNS-OARC,Nov. 2007.

[15] N/A, “Comparing DNS resolvers in the wild,” In IMC '10: Proceedings of the 10th annual conference on Internet measurement, Melbourne, Nov. 2010, pp. 15-21.

[16] F. Monrose, M. A. Rajab, and N. Provos, “Peeking Through the Cloud: Client Density Estimation via DNS Cache Probing,” ACM Transactions on Internet Technology (TOIT), Newyork, Oct. 2010, pp. 1-21.

[17] X. D. Li, B. P. Yan, X. C. Zhang, and W. Luo, “An IPv6 DNS Conformance Testing Framework,” In Computer-Aided Software Engineering, International Workshop, Jul. 2009, pp. 291-294.

[18] X. B. Yuchi, X. Wang, X. D. Li, and B. P. Yan, “DNS measurements at the .CN TLD servers,” The 6th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Aug. 2009, pp. 540-545.

[19] H. Nishimura, N. Maruyama, and S. Matsuoka, “Virtual clusters on the fly - fast, scalable, and flexible installation,” in CCGRID 2007: Seventh IEEE International Symposium on Cluster Computing and the Grid, May 2007, pp. 540-556.

[20] X. Y. Zhu, and D. Young, “An integrated approach to resource management for virtualized data centers,” Cluster Computing, Nov. 2008, pp. 45-47.

[21] X. Y. Wang, and Z. H. Du, “An autonomic provisioning framework for outsourcing data center based on virtual appliances,” Cluster Computing, Sep. 2008, pp. 229-245.