Cloud Computing

Assignment 2

Bhavesh Kumar Chigullapalli
S20190010034

Paper-1

Hussain, M., Wei, L. F., Lakhan, A., Wali, S., Ali, S., & Hussain, A. (2021). Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing. *Sustainable Computing: Informatics and Systems*, *30*, 100517
Link - https://www.sciencedirect.com/science/article/pii/S221053792100010X
Download the paper - paper-1

1. Motivation -

- **Energy reduction is one of the major concerns these days in virtualized cloud computing systems.**
- **Many fields such as signal processing, bioinformatics, Internet of Things(IoT) etc. use the services of cloud computing these days. These applications have millions of tasks performed by virtual machines at cloud data centers every day. But these virtual machines consume a lot of energy and this high amount of consumption raises the electricity costs and has a negative impact on the environment.**
- **Energy reduction offers us various benefits provided that we are still operating according to QoS, these benefits include reducing electricity costs, increasing system efficiency, and protecting the environment.**
- **An Energy efficient task scheduling algorithm is one of the ways to reduce energy consumption and promote green IT.**
- **A series of independent tasks were considered with the criteria of QoS such as deadline, difference size workload etc. But selecting a suitable virtual machine for each task to run is a difficult task as when you choose a poor performance virtual machine to run all tasks there will be queuing delay and QoS won't be achieved whereas if you choose to run them on a high performance virtual machine the energy consumed would be high.**

- **Hence the motivation to design an energy efficient task scheduling algorithm.**

2. Objective

- **The Objective of this paper is to design an energy efficient task scheduling algorithm that minimizes the energy consumption of the data centers while maintaining the Quality of service (QoS) terms intact.**

3. Major Contributions

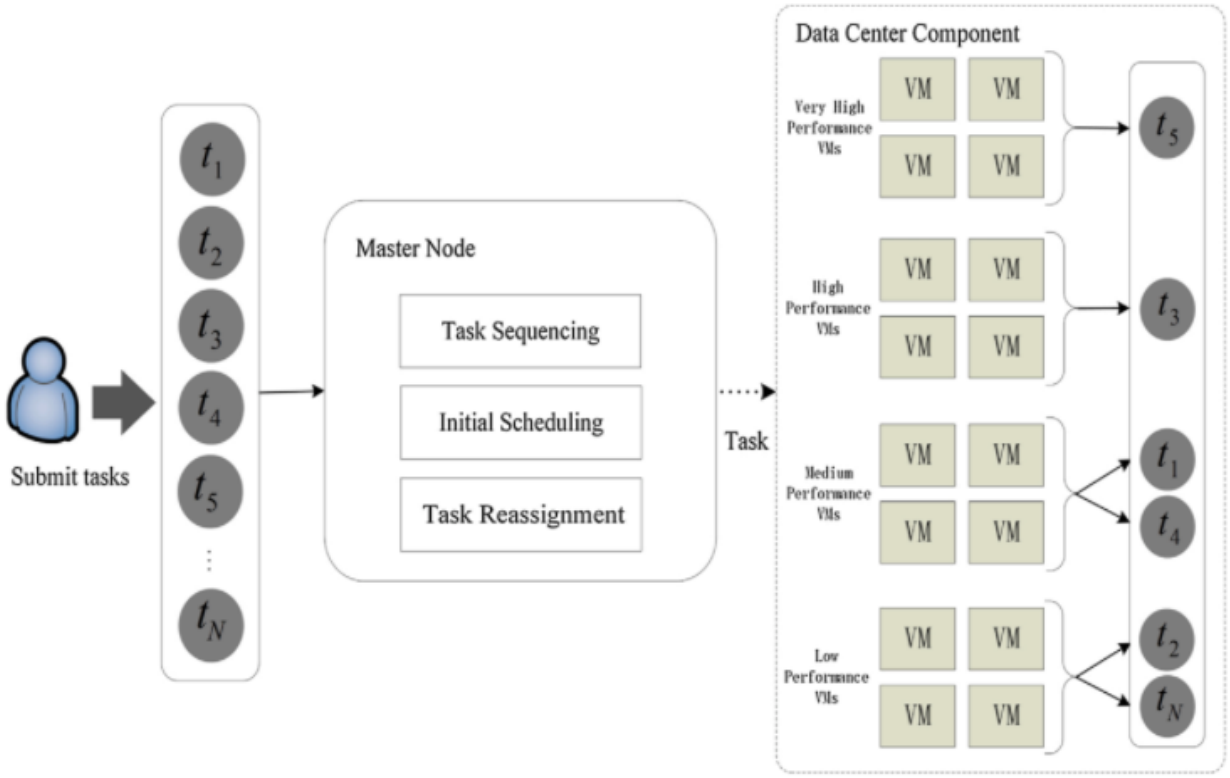
The paper has made the following contributions

- **This research paper provides a system architecture for processing tasks on heterogeneous computing virtual machines under tight deadlines using various components. Master node, job sequencing, initial scheduling, and energy-efficient task reassignment are the components. The system is made up of various virtual machines, and the scheduler will determine how and where jobs will be assigned based on the quality of service (QoS) criteria.**
- **This paper presents a unique algorithmic framework for solving the energy-efficient task scheduling problem. The framework includes Task sequence rules, initial scheduling, and task reassignment systems.**
- **The task sequence rule is the first step in the scheduling problem and the paper develops new rules which ensures the QoS during scheduling.**
- **This study has been experimented on the real test-bed in the simulations to measure the effectiveness and efficiency of the algorithm proposed.**

4. Proposed approach

Proposed Framework

- **The following figure shows the architecture of the framework and its components proposed by the paper.**



- The proposed architecture consists of three parts.
- Firstly the users send their independent tasks to the framework.
- These tasks are then sent to the second layer which is the management layer that consists of a master node that is responsible for the following tasks, task sequencing, initial scheduling and task reassignment.
    - Tasks submitted by users are sorted based on four task sequencing rules to simplify the scheduling operations,
    - Initial scheduling is done, where the scheduler depends on faster machines to reduce execution time and meet deadlines.
    - Then the scheduler reassigns the tasks to the slow machines from the faster ones to minimize energy consumption.
- Based on the algorithm the master node assigns the tasks to respective virtual machines and these are then carried out by the virtual machines responsible in the third part.
- Task characteristics - all tasks are independent, compute intensive and require resource intensive services to run them under QoS requirements.
- Resource characteristics - Virtual machines are the resources, this paper considers various types of virtual machines to handle all kinds of tasks while consuming minimal energy and also before deadlines.

Assumptions made by the paper for the algorithm:

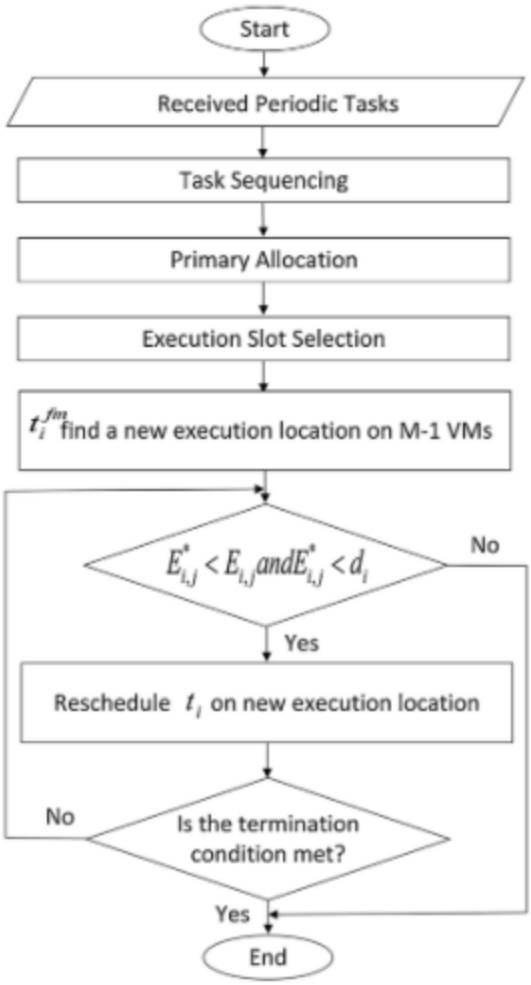
- **Tasks are independent.**
- **All the tasks are scheduled on the same data center, therefore there is no delay between the nodes.**
- **After the initial schedule, the task reassignment helps in reducing the energy consumption while maintaining the QoS requirements.**
- **All nodes are from same data center,**
- **Nodes do not fail but task assignment may fail and is taken into consideration.**

Energy consumption model:

- **Energy consumed by a virtual machine is mostly the energy consumed by the CPU.**
- **Let's denote E_{ij} as energy consumed by a virtual machine j, executing a task i, then our aim is to minimize $\sum E_{ij}$ over all tasks i=1 to N, where each task is executed before its deadline d_i.**
- **We achieve this by the following algorithm proposed in the paper.**
- **E_{ij}, that is, energy can be represented as "power consumed by virtual machine * execution time to complete a task", that is, energy = power * time, therefore energy is related to execution time and power of CPU in the virtual machines.**

Proposed Algorithm:

- **The following figure shows the working of the algorithm.**



- The EPETS(Energy and performance task scheduling) algorithm consists of two stages, initial scheduling and task reassignment.
- Initial scheduling is done in such a way that all tasks are executed before deadlines. In this way we ensure QoS.
- But this initial scheduling consumes considerable energy which is not optimal.
- To conserve energy we perform the second stage which is task reassignment.
- This paper proposes an energy efficient task priority framework to create a fair balance between energy consumption and task scheduling.
- Before each task is scheduled, the tasks are sequenced and then assigned to each VM.
- These tasks are generally sorted based on different sequence rules for example, deadlines, slack times etc.

- After the tasks are sorted, we do primary allocation, where we determine the tasks that are initially allocated to execute, we calculate minimum execution time of each task on the fastest VM and expected execution time on slower VM's.
- Based on the values calculated if estimated time on slower machine < minimum time on fastest machine then we assign such a task to slower machine.
- After calculating the values, we select and schedule tasks based on the priority. If a task is a slower machine task then we allocate it to a slower machine that has an available slot.
- We schedule tasks on faster and slower machines such that finish time is minimized and deadlines are met.
- After the allocation is done, we now move into the second phase of the algorithm which is task reassignment.
- The task reassignment algorithm is designed to reduce the cumulative energy consumption from all VM's.
- By reassigning tasks from faster VM's to slower or medium VM's energy consumption is reduced. Task reassignment from slower VM's to faster VM's is not considered in this algorithm.
- This algorithm works as an iterative algorithm where each task is considered for reassignment and is reassigned only if the reassignment conserves energy and also obeys QoS, and also the task must finish within deadline.
- Suppose there are N tasks to be reassigned and M virtual machines in total then there are at most N*M reassignment options. We only choose options that are optimal, that is, options that save energy and do not violate deadline rules.
- The previous step repeats until no more energy can be saved, and the task scheduling is finalized.

How this algorithm is different from existing algorithms:

- This paper also considered task slack time calculation which helps to improve the system performance, reduce energy consumption and deadline violations
- The algorithm is divided into two stages, in the first stage we schedule all the tasks to meet SLA requirements and in the second stage we use task reassignment to find optimal assignment that minimizes the energy consumption.

Time Complexity analysis:

- The time complexity of initial scheduling algorithm is O(N * logn), where N is the number of tasks, and logn is for sorting iteration of all tasks
- The time complexity of task reassignment algorithm is O(N*M) where N is the number of tasks and M is the number of virtual machines in the data center.
- Total time complexity is O(Nlogn + N*M)

5. Experimental testbed and results

- A set of randomly generated tasks is used to test the efficiency and energy consumption of the EPETS algorithm with 3 other existing algorithms namely, AMTS, RC-GA, E-PAGA.
- In this experiment 5 different task nodes were generated, and relative percentage deviation was used to differentiate the performances of algorithms, note that the task nodes are of 5 different types and the total number of task nodes are not 5, they might be more than 5.
- 4 different types of virtual machines are used based on amazon EC2, each VM has different power and processing speed and can be classified as slower, medium and faster VM's.
- Task sequence rules considered are first come first service and earliest deadline first.
- The experiments have shown that the EPETS algorithm outperforms all the 3 existing algorithms in both performance and energy saved.

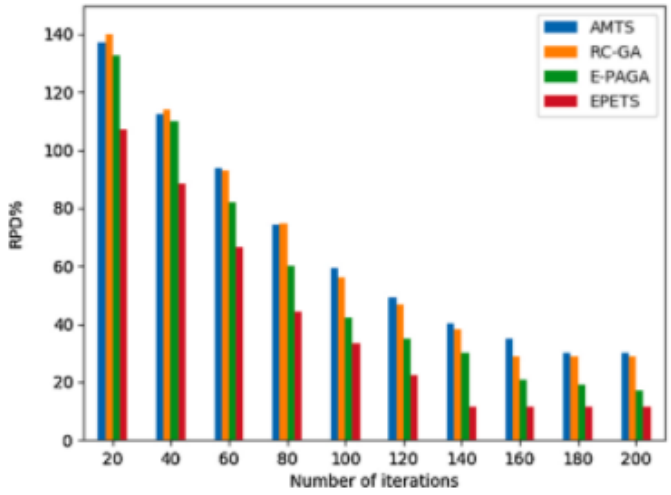


Fig. 8. Total execution time with iteration number.

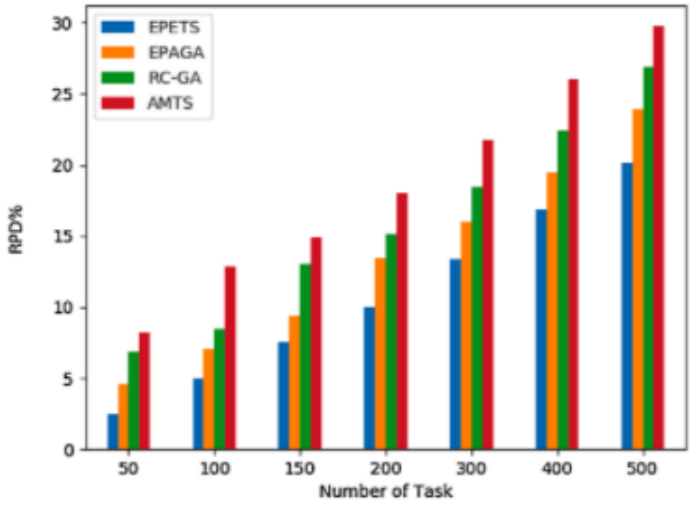


Fig. 11. Energy consumption with task number.

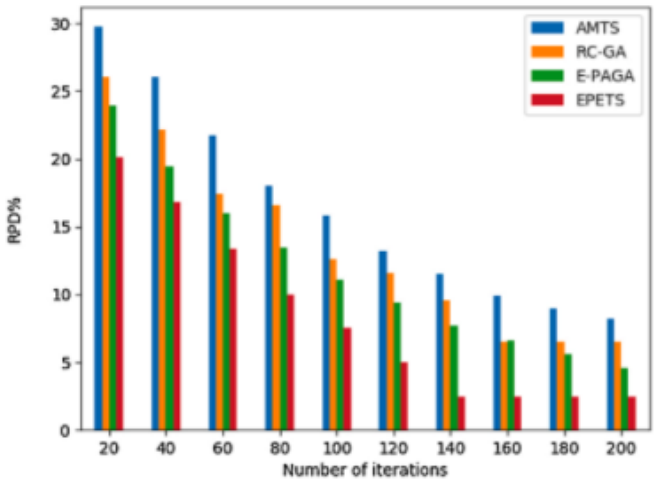


Fig. 12. Energy reduction with iteration number.

- The charts above clearly show that the EPETS algorithm successfully reduced more energy consumption and also improved performance while not violating the QoS.

6. Strengths

- The proposed solutions help in significantly reducing energy consumption while not violating the quality of services requirements.
- Also the results have proven that the algorithm successfully achieves what it aims for and significantly reduced energy consumption compared to the existing algorithms.
- Also the algorithms improved the performance of the virtual machines and the resource utilization, there is about 5%-20% of improvement in performance while deadline constraints are satisfied.
- This algorithm helps conserve a considerable amount of energy for each data center and when multiple data centers follow the same approach huge amounts of energy can be saved and this helps in green IT.

7. Limitations -

- Failure of virtual machines are not taken into consideration - In case a virtual machine failure occurs due to some random reason, then the tasks assigned to that particular machine needs to be reassigned. This raises a new problem.
- The paper disregarded intercloud and multi cloud environments and only considered that the nodes are from the same datacenter.

Paper-2

Lee, J. B., Yoo, T. H., Lee, E. H., Hwang, B. H., Ahn, S. W., & Cho, C. H. (2021). High-Performance Software Load Balancer for Cloud-Native Architecture. *IEEE Access*, *9*, 123704-123716.
Link - https://ieeexplore.ieee.org/abstract/document/9524915
Download the paper - paper-2

1. Motivation

- **The demand of cloud computing has been increasing lately and cloud providers are always trying to install a reliable and easy to manage cloud architecture.**
- **Cloud software components can be packaged together to form a container and this container can be managed using orchestration tools such as the kubernetes engine.**
- **Similarly load balancers can also be deployed in containerized cloud environments and managed as a container.**
- **This enables us to automatically scale the cloud resources, and high performance and efficient load balancer is needed for proper utilization of resources and to provide best services.**
- **CSP's usually package components and form a container to reduce OS dependencies that arise when a new data center is established, thus increasing the need of high performance load balancers.**

2. Objective

- **The objective of this paper is to build a containerized high performance load balancer that can be easily managed using orchestration tools such as kubernetes engine in a cloud native architecture.**
- **A cloud-native architecture refers to an environment in which each component is packaged and can be deployed independently on any infrastructure.**

3. Major Contributions

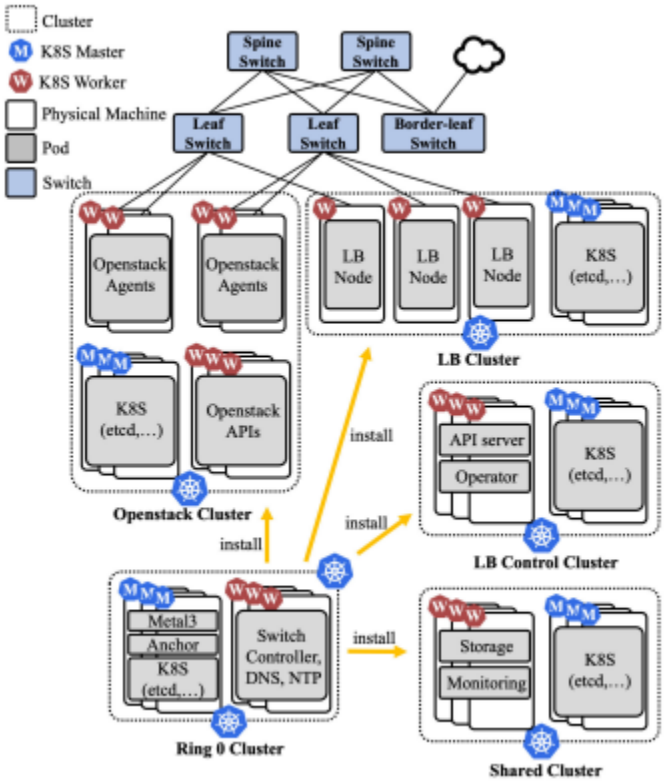
- **Define a cloud native architecture for deploying the load balancer proposed that runs on this environment.**
- **Present the load balancer(L2DSR) architecture that can be easily managed with kubernetes.**
- **Implement a eBPF/XDP-based high performance load balancer.**

- The load balancer proposed supports various network protocols which enables a multi-tenant environment.

4. Proposed Approach

Cloud native Architecture with load balancer

- Kubernetes Clusters
    - The following figure shows how each cluster is created, the roles of each cluster and which pods are deployed to the related clusters.



    - Only Ring0, openstack shared clusters and LB clusters are used.
    - For each server in the data center one port is connected to the leaf switch and another port to the leaf switch of the management network.
    - Ring0 cluster: This cluster installs and manages other clusters in the network and itself as well. Three open source projects were used to create clusters and manage them, Ring0 cluster is responsible for creating and managing clusters including itself.

    - Shared cluster: This cluster contains applications related to storage and monitoring. Services that run commonly on all servers are deployed and run on the shared cluster.
    - Openstack: Openstack clusters operate in two groups, in the first group API servers such as octavia,Neutron etc are operated as pods, and the second group contains VM's created by users.
    - LB cluster: This cluster contains pods that are load balancers. Herein the pods are referred to as LBN's (load balancer nodes). Each worker node in the LB cluster runs only one LBN and in case a new node is added, a new pod is also created and the new node will run on the new pod created. Similarly if a node is deleted, the pod on which the node ran is also deleted.
    - LB control cluster: This cluster contains API server and LB operator that acts as controller LBNs located in LB cluster.
- Multi Tenant networks
    - For multi-tenant networks, networks of different tenants are segmented based on their VLAN ID (VID), even if VM's of different tenants have the same IP they can be differentiated using VID.
    - A VLAN header is appended to a packet generated in a compute node, along with the VID of the relevant network, when it exits the node.
    - A VXLAN header is appended to a packet instead of a VLAN header because the leaf-spine fabric is configured with VXLAN-EVPN.
    - A VXLAN tunnel endpoint (VTEP) is the start/end point of a VXLAN tunnel on each leaf switch.
    - The original user data frames are thus encapsulated and decapsulated by each leaf switch. VTEP maps a given VID to a specific VNI value of VXLAN in a leaf switch. Because each leaf shares the learned MAC addresses through VXLAN-EVPN, traffic flooding to all of the leaf switches is reduced.
    - This reduces the amount of broadcast, unknown-unicast, and multicast (BUM) traffic in the network, which eliminates a variety of difficulties that can arise as a result of a broadcast storm.
- Load balancer setup procedure
    - LBNs are assigned group numbers initially when they are created and they operate in groups of finite numbers that can be scaled up or down based on the incoming traffic, these LBNs together handle the same VIP.

- The external router distributes traffic evenly across each LBN in a group through equal cost multi path (ECMP) protocol.
- The following figure shows components associated with LBNs and scenarios in which these LBNs are deployed in the cloud.

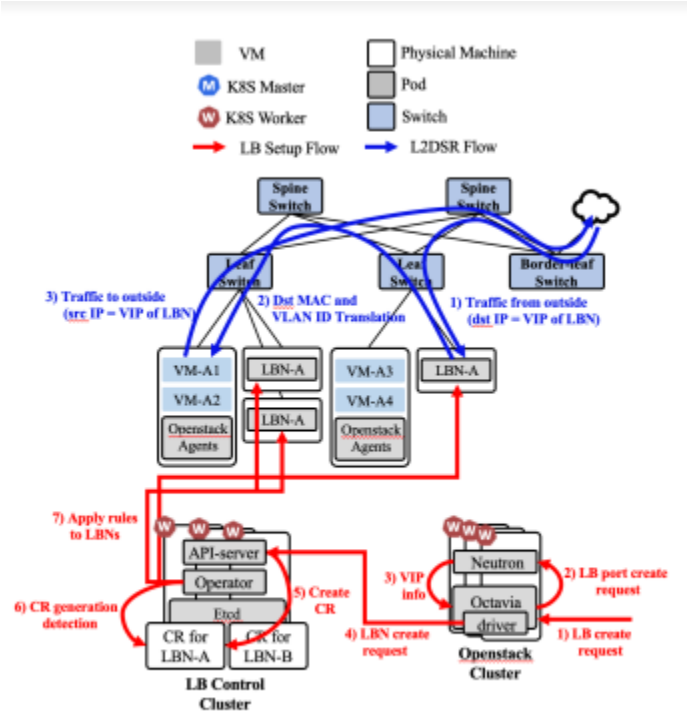


FIGURE 2. Process of setting up LBNs and L2DSR by LBN.

- LB control path: openstack cluster receives LB create req, it creates a port for LB to use and returns VIP to be used by the LB. The API server creates, modifies and removes a custom resource based on predefined rules for a LBN group.
- LB data path: In the above figure, we can observe how LAB-A in group A distributed load to VM-A1,VM-A2,VM-A3,VM-A4. A Hashing algorithm is used to assign the load to a particular virtual machine.

Implementation of LBN:

- LB is created in the form of a container and is replaced with a part of the physical LB used in the production data center.
- LBN core is responsible for dividing the load while LBN controller assists LBN core.
- LBN controller aims to inject the program of LBN core and update hash tables that can be used by LBN core.

- The paper proposes two types of hash tables, in the first hash table the result is the physical address of the VIP, second hash table is composed of backend VM information and these hash tables are maintained in BPF maps.
- The LBN controller updates, overwrites, modifies the existing hash tables, LBN core and controller share the same hash tables.
- Maglev hashing algorithm is used for hashing to minimise hashing disruptions that occur.

5. Experimental testbed and results

Experiment setup:

- The experimental environment used is as RFC2544 suggests, there are two servers connected to the same switch where one server is the tester and the other is the one being tested.
- The tester has TRex installed which is an open source traffic generator.
- Each server is configured with 2-10GHZ CPUs with 16 total cores and four 32GB RAM units were used. UBUNTU 20.4 operating system with linux kernel 5.4 was installed on each server.
- RFC2544 performance test, IMIX performance test and LBN deployment test have been performed.

Testing and results:

- RFC2544 performance test: seven test frame sizes were used (64,128,256,512,1024,1280,1510). Four types of tests were conducted for each frame size specified: loopback, iptables DNAT, scenario-A = (#VIP=1 and #REAL-VM=255) and scenario-B = (#VIP=128 and #REAL_VM=4000) . UDP and TCP traffic was generated. It was observed that smaller the size of the frame increases the overhead of handling frames. At smaller sizes of frame LBN performed less than what was expected and in comparison with loopback and iptables DNAT, but at higher frames LBN performed better and remained constant. Also it was determined that the number of rules do not affect the performance of LBN at all.
- IMIX performance test: LBNs performance was approximately 2% lesser than theoretical maximum performance, but the performance of the proposed LBN was 27 times better than iptables DNAT. The number of rules did not affect the performance of the LBN.

- LBN Deployment test: The preparation time required for LBN after deployment was less than 1s and it took approximately 3s for a general server to turn into high performance load balancer.
- The time required to apply rules made in Openstack cluster's octavia was 3s on average and most of the time was identified as time required for octavia, neutron etc in openstack.

6. Strengths

- Introduced an installable cloud that enables IaaS services.
- Utilizes open source solutions such as kubernetes, cluster API, Openstack.
- The experimental results indicate that the throughput of the LBN is significantly greater than the iptable DNAT and the performance increases as the packet size increases.
- The difference between loopback and LBN was minimal so the proposed LBN can be used in commercial applications.

7. Limitations

- The proposed LBN worked well for packets with higher frame sizes but had a poor performance with lower packet sizes, according to experiment results LBN performed 24% lesser than optimal performance for frame size of 64 and 3% lesser for a frame size of 128 although the performance remained constant and better for higher frame sizes.
- The proposed LBN contains multiple clusters with different motives, but altogether gives the execution of load balancing, so failure of one component endangers the performance of LBN.