Report Organization

- Chapter 1: **Introduction.**
- Chapter 2: **Virtualization.**
- Chapter 3: **Cloud.**
- Chapter 4: **Prerequisites to Implement the Project.**
- Chapter 5: **Migration.**
- Chapter 6: **Load Balancing.**
- Chapter 7: **Proposed Algorithm.**
- Chapter 8: **Implementation of Algorithm and Development of GUI.**
- Chapter 9: **Performance and Result Analysis.**
- Chapter 10: **Conclusions and Future Work.**
- **References.**

Table of Contents

CHAPTER I
INTRODUCTION

Virtualization is commonly defined as a technology that introduces a software abstraction layer between the hardware and the operating system and applications running on top of it. Its main advantages include isolation, consolidation and multiplexing of resources. Other benefits of virtualization include saving on power by consolidation of different virtual machines on a single physical machine, migration of virtual machine for load balancing etc. Virtualization provides full control of resource allocation to administrator, resulting in optimum use of resources. More recently, another advantage of virtualization - live migration of virtual machine is increasingly used to better handle workload balancing across physical machines in data center, especially when the available resources in physical machine are not sufficient for VMs.

The load balancing problem determines how to divide work between available machines in a way that achieves performance goals. In static load balancing, the mapping of jobs to resources is not allowed to change after the load balancing has begun. On the other hand, dynamic load balancing will change the mapping of jobs to resources at any point, but there may be a cost associated with the changes. Load balancing is essentially a resource management and a scheduling problem. The operating system on each host performs local scheduling. Local scheduling involves deciding which processes should run at a given moment. Global scheduling, on the other hand, is the process of deciding where a given process should run. Global scheduling may be performed by a central authority or it can be

distributed among all the hosts. The goal of the proposed work is to dynamically balance load on the hosts by a central authority, which is also a virtual machine.

Chapter II gives an overview about Virtualization and live migration and other prerequisites for the setup. Chapter VI explains load balancing concept and phases involved in it. Chapter VII proposes the algorithm for dynamic load balancing based upon the framework given in Chapter VI. Results are shown in Chapter IX. Finally, some conclusions are drawn in Chapter X.

CHAPTER II
Virtualization

Core of any virtualization technology is Hypervisor .Hypervisor is a piece of software which allows each virtual machine to access and schedule the task on resources like CPU, disk, memory, network, etc. At the same time hypervisor maintains the isolation between different virtual machines. Virtualization can be classified by the method in which hardware resources are emulated to the guest operating system. They are as follows:

2.1 Full Virtualization:

Hypervisor controls the hardware resources and emulates it to guest operating system. In full virtualization, guest does not require any modification. KVM is an example of full virtualization technology.

2.2 Paravirtualization:

In paravirtualization, hypervisor controls the hardware resources and provides API to guest operating system to access the hardware. In paravirtualization, guest OS requires modification to access the hardware resources. Xen is an example of paravirtualization technology.

2.3 Back Bone of Cloud Computing:

Virtualization is the key role to cloud computing, since it is the enabling technology allowing the creation of an intelligent abstraction layer which hides the complexity of underlying hardware or software.

Server Virtualization enables different operating system to share the same hardware and make it easy to move operating system between different hardware, all while the application are running

Cloud can exist without Virtualization, although it will be difficult and inefficient.

Cloud makes the notation of “Pay for what you use”, Infinite availability-use as much you want.

These notation are practical only if we have

1. Lot of flexibility

2. Efficiency in the back-end.

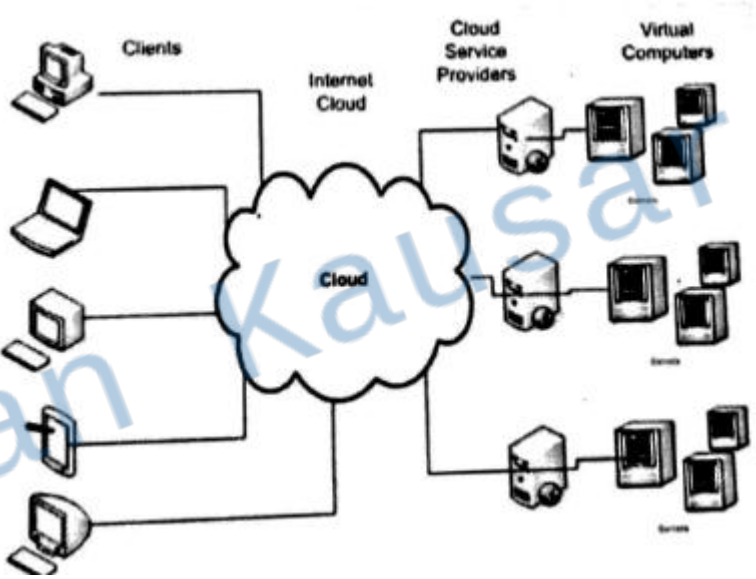
These efficiency is readily available in Virtualized Environment and Machine.

CHAPTER III
Cloud

Came into Existence around 1950, The underlying concept of cloud computing dates back to the 1950s, when large-scale The underlying concept of cloud computing dates back to the 1950s, when large-scale mainframe computers became available in academia and corporations, accessible via thin clients/terminal computers, often referred to as "static terminals", because they were used for communications but had no internal processing capacities. To make more efficient use of costly mainframes, a practice evolved that allowed multiple users to share both the physical access to the computer from multiple terminals as well as the CPU time. This eliminated periods of inactivity on the mainframe and allowed for a greater return on the investment.

"The **cloud** itself is a set of hardware, network, storage, service and interface that enables the delivery of computing as a service. **Cloud service** include the delivery of software, infrastructure and storage over the internet (Either as a separate components or a complete platform) based on user demand."

Cloud computing gets its name as a metaphor for the Internet. Typically, the Internet is represented in network diagrams as a cloud as shown in the figure 1 The cloud icon represented "All that other stuff" that makes the network work. It's kind of like "etc." for the rest of the solution map.



(Figure:-1)

It also typically means an area of the diagram or solution that is someone else's concern. It's probably this notation that is most applicable to the cloud computing concept. Parallels to this concept can be drawn with the electricity grid where end-users consume power resources without any necessary understand of the component device in the grid required to provide the service.

3.2 Cloud Deployment Models:

3.2.1 Public Cloud

A public cloud is one based on the standard cloud computing model, in which a service provider makes resources, such as applications and storage available to the general public over the Internet. **Public cloud services may be free of offered on pay-per-usage model.**

The main benefits of using a public cloud service are:

- Easy and inexpensive set-up because hardware, application and bandwidth costs

are covered by the provider

- Scalability to meet needs.
- No wasted resources because you pay what you use.

3.2.2 Private Cloud

Private cloud (also called internal cloud or corporate cloud) is a marketing term for proprietary computing architecture that provides hosted service to a limited number of people behind a fireball. In other words, it refers to having your own private cloud computing infrastructure.

So, instead of relying on an external public cloud service provider's infrastructure you would have your own. A private cloud is more suited for a large enterprise because it has already invested heavily in its IT infrastructure, so that it deliver service to the users faster and more effectively. It provide better control over the entire process of information processing.

3.2.3 Hybrid Cloud:

The term "Hybrid cloud" has been used to mean either two separate clouds joined together (public, private, internal or external), or combination of virtualized cloud server instances used together with real physical hardware and virtualized cloud service instances together to provide a single common service. Two cloud that have been joined together are more correctly called "combine cloud"

3.2 Cloud Computing Models:

Whether it's public, private or hybrid different range of service could run in cloud computing environment. They are divided into three parts:

3.3.1 Software as a Service (Saas).

These services are typically available over the public Internet and are information-based. This category of cloud computing is the most mature and best known.

3.3.2 Platform as a Service (PaaS).

These services provide an application development platform that enables application authoring in a runtime environment, without the hardware investment. This is the most recent entry as a cloud computing service model, and it offers the promise of rapid software development without the need to provision or configure infrastructure for the application.

3.3.3 Infrastructure as a Service (IaaS).

These services provide IT infrastructure that can be deployed and used via remote Access, providing virtual hardware that can be deployed via programmatic methods or from the command line. The Iaas customer rents computing resources instead of buying and installing them in their own data center.

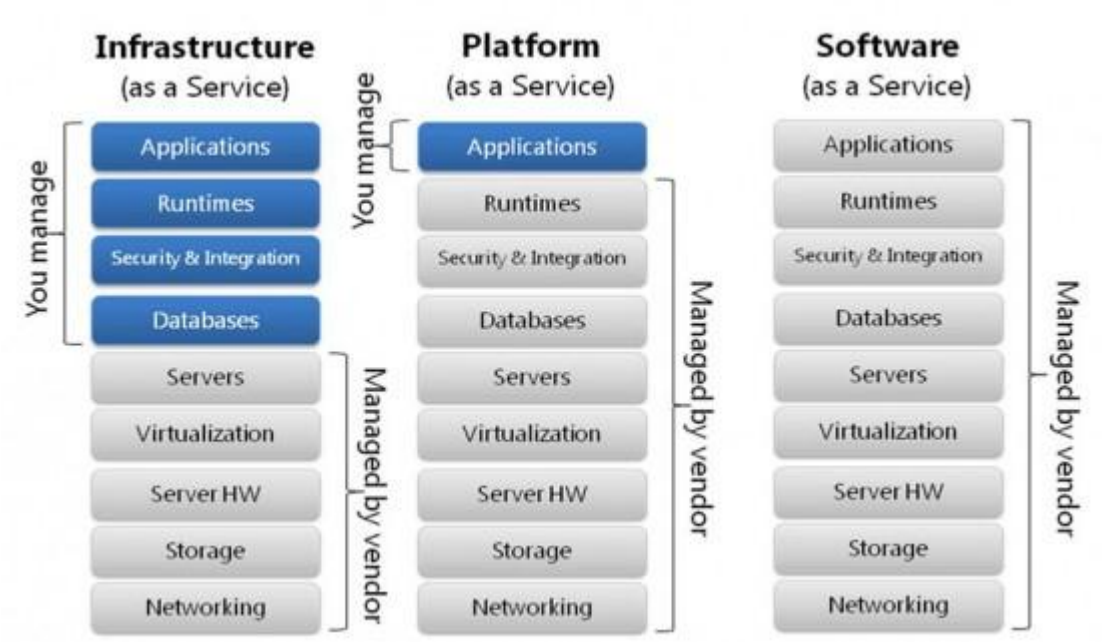


Figure2:- Cloud Computing Models.

CHAPTER IV
Prerequisite to Implement the Project

4.2 Hardware:

Operating System	**CentOS7**
CPU	**Intel Core i3-2400 3.10 GHz * 4**
Memory	**3 GB.**
Storage	**500 GB**

Table1:-Hardware Details.

4.2 Software:

4.2.1 KVM

Kernel-based Virtual Machine (KVM) project represents the latest open source virtualization technology. KVM is implemented as a loadable kernel module that converts the Linux kernel into a bare metal hypervisor. In the KVM architecture, the virtual machine is implemented as regular Linux process, scheduled by the standard Linux scheduler. In fact each virtual CPU appears as a regular Linux process. This allows KVM to benefit from all the features of the Linux kernel.

4.2.2 QEMU

Q-Emulator is a generic open source processor emulator and virtualizer. It can run many Operating Systems and programs made for one machine on another. It uses dynamic binary translation to achieve high performance. Binary Translation is an emulation technique in which, instead of emulating the processor, the virtual machine runs directly on the CPU .

4.2.3 Shared Storage using Network File System (NFS)

The hosts in a live migration write to the same VM image file when a VM is moved. Thus, Live migration requires shared storage which is provided by NFS.

CHAPTER V
Migration

5.1 What is Migration?

Moving of a Virtual Machine (VM) from one host to another. It is use in Load Balancing, Maintenance, and Recovery from Host failure.

5.1 Types of Migration.

Cold Migration: - In this type of migration first VM is shutdown on the source node and then it is restart on destination node.

Warm Migration: - Here VM is suspended on source node and copy across RAM and CPU register, continue on destination node

Live Migration: Live Migration [6] of a virtual machine is simply moving the running VM on a physical machine (source host) to another physical machine (target host) without disrupting any active network connections, while the VM is running on the source host, even after the VM is moved to the target host. It is considered live, since the original VM is running, while the migration is in progress. Very small downtime, in the order of milliseconds, is the benefit of doing live migration. We can migrate a guest between an AMD host to an Intel host and back. Naturally, a 64-bit guest can only be migrated to a 64-bit host, but a 32-bit guest can be migrated to 32 or 64 bit host.

5.3 Live Migration Steps.

The live migration task comes down to the following steps.

5.3.1 Pre-Migration.

Select VM to be migrated and destination host where resources required are guaranteed to be present.

5.3.2 Reservation.

In this stage, confirmation of necessary resources at destination host is done and a VM container of that size is reserved.

5.3.2 Iterative Pre-Copy.

The pages from guests physical address space are sent one at a time to the destination system i.e. the guest's memory is copied to the destination. The pages that are copied are made read-only for the guest during the live migration process. In the following iterations, only the dirtied pages are copied.

5.3.4 Stop and Copy.

At this stage, VM at source host is suspended and network traffic is redirected to destination host. Also CPU state and any remaining inconsistent memory pages are then transferred.

5.3.5 Commitment.

Now, Destination host indicates source host that it has successfully received a consistent VM image.

5.3.6 Activation. The migrated VM on destination host is now activated.

CHAPTER VI
Load Balancing

6.1 What is Load Balancing?

Load balancing in the cloud differs from classical thinking on load-balancing architecture and implementation by using commodity servers to perform the load balancing because it's difficult to predict the number of requests that will be issued to a server. This provides for new opportunities and economies-of-scale, also presenting its own unique set of challenges. Load balancing is one of the central issues in cloud computing.

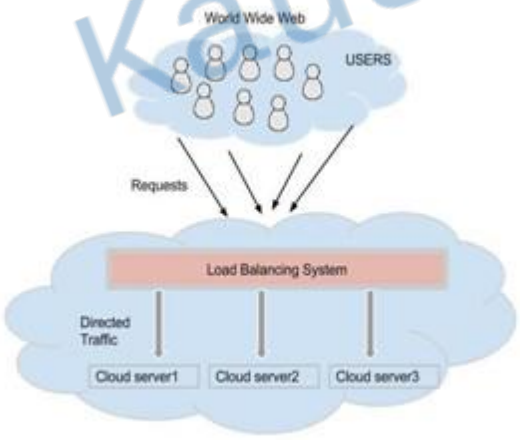


Figure 3:- Load Balancing system in cloud computing

Load balancing is the process of reallocating VMs on another host in the network in order to improve resource and network utilization.

Efficient Provisioning of resources and scheduling of resources as well as tasks will ensure:

a. Resources are easily available on demand.

b. Resources are efficiently utilized under condition of high/low load.

c. Energy is saved in case of low load (i.e. when usage of cloud resources is below certain threshold).

d. Cost of using resources is reduced.

6.2 Need of Load Balancing in Cloud Computing

Load balancing in clouds is a mechanism that distributes the excess dynamic local workload evenly across all the nodes. It is used to achieve a high user satisfaction and resource utilization ratio [15], making sure that no single node is overwhelmed, hence improving the overall performance of the system. Proper load balancing can help in utilizing the available resources optimally, thereby minimizing the resource consumption. It also helps in implementing fail-over, enabling scalability, avoiding bottlenecks and over-provisioning, reducing response time etc. Apart from the above-mentioned factors, load balancing is also required to achieve Green computing in clouds which can be done with the help of the following two factors:

• **Reducing Energy Consumption** - Load balancing helps in avoiding overheating by balancing the workload across all the nodes of a cloud, hence reducing the amount of energy consumed.

• **Reducing Carbon Emission** - Energy consumption and carbon emission go hand in hand. The more the energy consumed, higher is the carbon footprint. As the energy consumption is reduced with the help of Load balancing, so is the carbon emission helping in achieving Green computing.

6.3 Types of Load Balancing.

6.3.1 Centralized Load balancing.

In centralized load balancing technique all the allocation and scheduling decision are made by a single node. This node is responsible for storing knowledge base of entire cloud network and can apply static or dynamic approach for load balancing. This technique reduces the time required to analyze different cloud resources but creates a great overhead

on the centralized node. Also the network is no longer fault tolerant in this scenario as failure intensity of the overloaded centralized node is high and recovery might not be easy in case of node failure.

6.3.2 Distributed Load Balancing.

In distributed load balancing technique, no single node is responsible for making resource provisioning or task scheduling decision. There is no single domain responsible for monitoring the cloud network instead multiple domains monitor the network to make accurate load balancing decision. Every node in the network maintains local knowledge base to ensure efficient distribution of tasks in static environment and re-distribution in dynamic environment. In distributed scenario, failure intensity of a node is not neglected. Hence, the system is fault tolerant and balanced as well as no single node is overloaded to make load balancing decision.

6.3 Goal of Load Balancing.

Following are the goals of load balancing :

- To improve the performance substantially
- Fault tolerance in case of system failure
- To maintain the system stability
- To accommodate future modification in the system

6.4 Types of Load Balancing Algorithm.

Following are types of load balancing algorithms:

- Sender initiated: Algorithm initiated by Sender.
- Receiver initiated: Algorithm initiated by Receiver.
- Symmetric: Combination of above two.

CHAPTER VII
Proposed Algorithm

7.1 Introduction of Proposed Algorithm.

The proposed algorithm is modified version of Central Scheduler Load Balancing (CSLB) algorithm. The algorithm uses the four phases for load balancing.

Phase1: Load Evaluation.

Phase2: Profitability Determination.

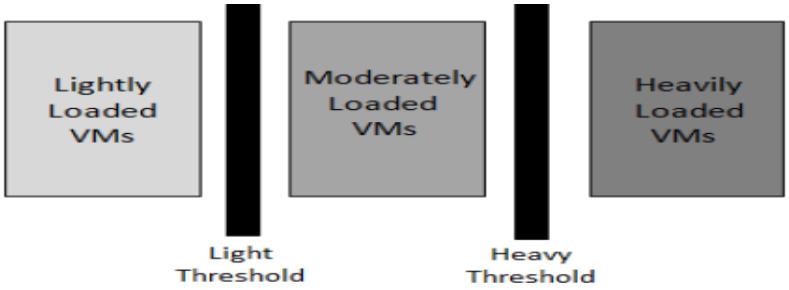
Phase3: Task (VM) Selection.

Phase4: Task (VM) Migration.

7.2 Phases of Algorithm.

7.2.1 Load Evaluation.

As stated, the algorithm is a central scheduling algorithm, where all the hosts send load information to the policy engine: which is responsible for load balancing decisions, after a predefined time interval which can be changed as per the requirements.



(Fig4: CPU Utilization Band)

The CPU utilization is divided into three bands: Lightly loaded, moderately loaded and heavily loaded based on the threshold value as shown in above figure.

The threshold is the average of CPU usage of all hosts. The moderately loaded band is created by adding and subtracting a value, which is the difference between the average and

the mean of the maximum and minimum of the CPU usage of the hosts or of 20 percent width, across the threshold, whichever is maximum. All the hosts fall in at least one of the bands. The systems would be in an ideal state if all the hosts lie in the moderately loaded band.

Calculation of moderate band:

threshold = *(α1+α2+α3+...+αn)/n*

α close= *α i whose CPU utilization is close to threshold.*

α max*= α i whose CPU utilization is Maximum of all.*

Mean*= (α close + α max)/2.*

Diff*= | threshold – Mean |*

If diff > = 20 % of threshold

moderately loaded band *= threshold ± diff*

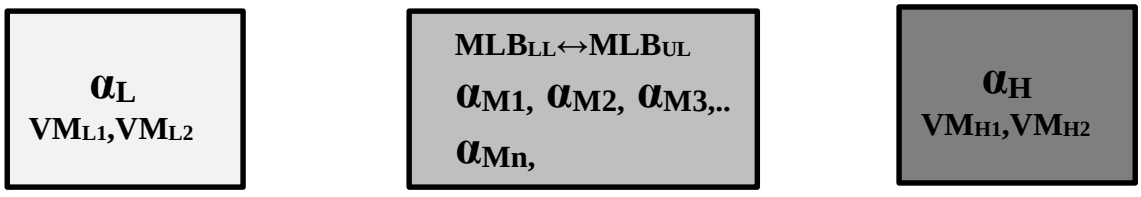
else

moderately loaded band *= threshold ± 10*

(where, αi= CPU usage of ith Host over a defined time in percentage and i = 1 to n)

7.2.2 Profitability Determination.

Migration of virtual machines from one host to another will be considered profitable if there exists one virtual machine in the heavily loaded band and one in lightly loaded band.



(Figure 4: Profitability determination)

(L=Lightly Loaded Band, M=Moderately Loaded Band, H=Heavily Loaded Band.

LL=Lower Limit, UL=Upper Limit.)

7.2.3 Task (VM) Selection.

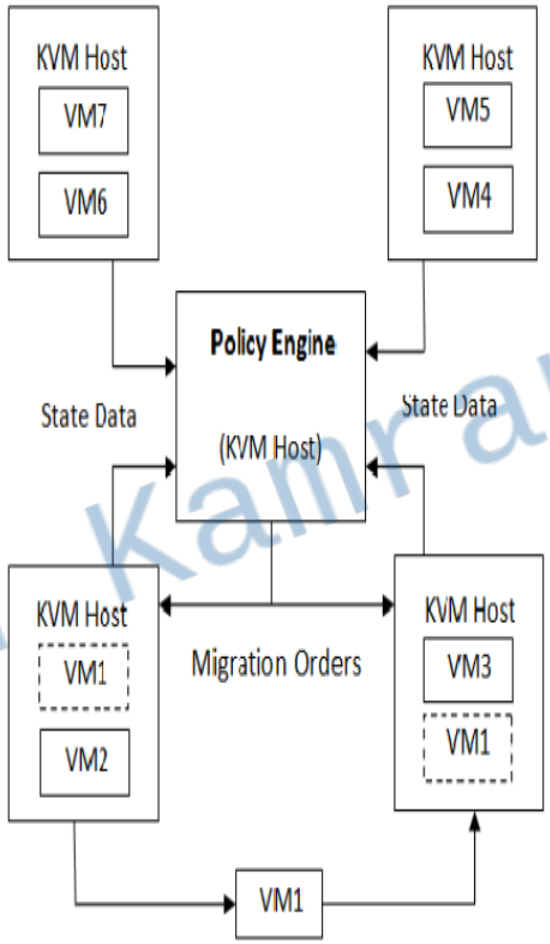
VM Selection can be estimated using the virtual machine usage on a host. The virtual machine that has a usage closer to node which comes under lower limit of Moderately Loaded Band. i.e.

VM_{Hi}(Select)= VM_{Hi} close to MLB_{LL} (Moderately Loaded band Lower Limit**)**

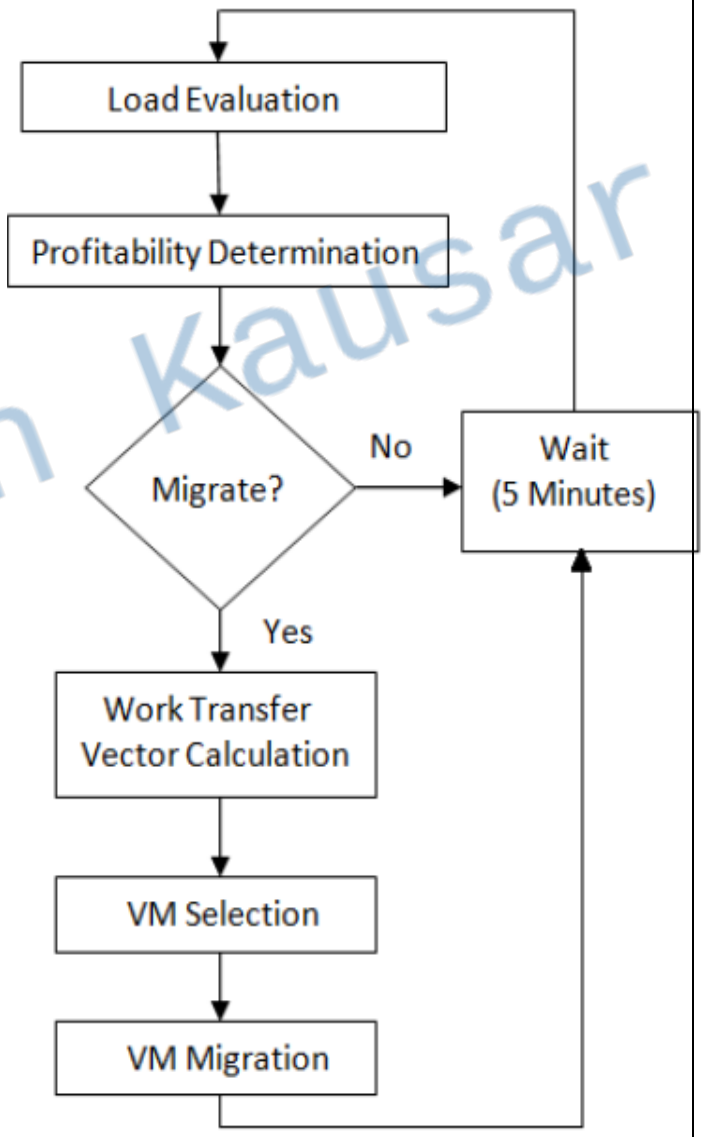
NOTE: In scenarios, where there are multiple nodes in Low Band and multiple nodes in High Band. This improved algorithm automatically transfers the maximum "work" possible. This is an enhancement to the original algorithm.

7.2.4 Task (VM) Migration.

The Virtual machine that is selected in Phase 3 should be migrated. Here QEMU-KVM's live migration feature is used. It is done using the virsh command which is a libvirt API (a toolkit to interact with Virtualization capabilities of OS). The relation between the phases of the algorithm is explained by the following flowchart (Fig 6) and (Fig 7):



(Fig 6: Relationship between policy engine and KVM hosts.)



(Fig 7: Flow Chart of Phases of Algorithm)

CHAPTER VIII
Implementation of Algorithm and Development of GUI.

8.1 Project Codes

8.1.1 Implementation of Algorithm.

Implementation of Algorithm is done by using C Language and Bash scripting. To store all the real time information of Nodes and VMs details a separate Database is maintained in CSV (Comma Separate Value).

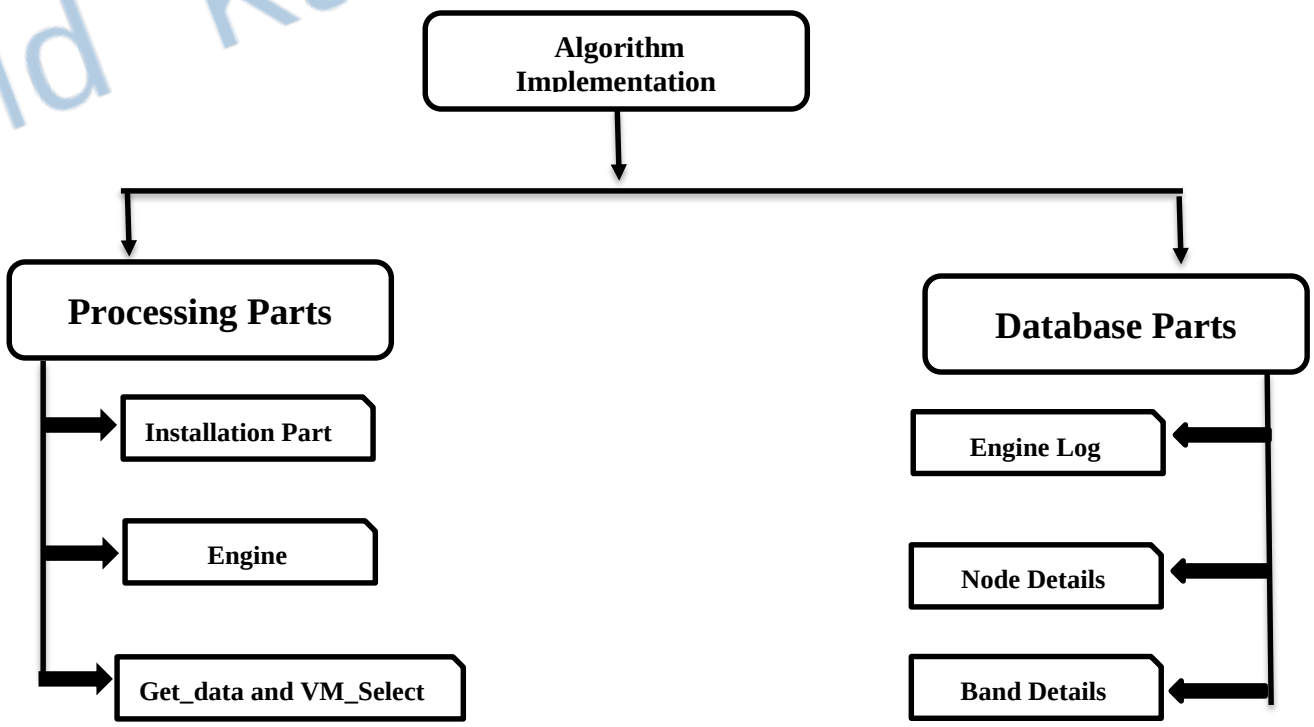


Figure6: Software Project Development Steps

Important Processing Part Code:

//Installation Part

clear

echo

```
echo "|###################### INSTALLER FOR ######################|"
echo "|##################DYNAMIC LOAD BLANCING ###################|"
echo "|#####################  ON  ####################################|"
echo "|###############| LINUX PRIVATE CLOUD ########################|"
echo
echo "  Here are the Pre-Requisites to implement this Project"
echo -e "\nThese are\n
        1. Machine support virtualization features i.e one of the CPU flag is VMX or
SVM
        2. Installation of KVM.
        3. Installation of QEMU.
        4. Passwordless SSH configured between  Server and Client node.
        5. Libvirtd TCP remote URI 16501
        6. All nodes have configured Shared storage by NFS."
echo    "Are Your Pre-Requisites configured Correctly? Enter Y/N"
read a
        if [ "$a" == "y" ] || [ "$a" == "Y" ]
        then
echo    "Enter the number node in your  Cluster on which this has to be implement"
        read node_total
        echo    "Enter the name of the node One by One"
        c=0
        cat /dev/null>./data/nodeinfo
```

```
while [ $c -lt $node_total ]
        do
        read nodename
        echo -e $nodename >>./data/nodeinfo
        c=`expr $c + 1`
done
echo
echo "Node names are  successfuly Enter"
echo
else
        cat /dev/null>./data/nodeinfo
        echo    "Quitting!!! " System is not configured Correctly""
        echo
fi
#Housekeeping
#clear
echo "Start time :"; date

echo "####### Cleaning up last run data ########"
echo "done!"

cat /dev/null>./data/engine.log
cat /dev/null>./data/node_cpu.log
```

```
c=0;thrs=0;max=0;min=100;diff=0;
./getdatav3.sh &
echo "######## Retrieving Private Cloud Performance Data (Hosts and their VMs) ###"
##Getting Host Performance Data
for i in `cat ./data/nodeinfo`
        do
        cpu=$(ssh root@$i mpstat 1 10|tail -1 |awk -F" " '{print 100 - $11}')
        echo $i,$cpu >>./data/node_cpu.log
        thrs=`echo $thrs + $cpu | bc -l`
        c=`echo $c + 1|bc`
        if [ $(echo "$max < $cpu"|bc) -eq 1 ]; then max=$cpu; fi
        if [ $(echo "$cpu < $min"|bc) -eq 1 ]; then min=$cpu;fi
done
thrs=`echo "scale=2; $thrs / $c"|bc`
mean=`echo "scale=2;($max + $min)/2"|bc`
if [ $(echo "$thrs > $mean"|bc) -eq 1 ];
        then
        diff=$(echo "scale=2;$thrs - $mean"|bc);
 else
        diff=$(echo "scale=2;$mean - $thrs"|bc);fi
        if [ $(echo "$diff < 10"|bc) -eq 1 ];then diff=10;fi
        if [ $(echo "$thrs < $diff"|bc) -eq 1 ]; then thrs_low=0;else thrs_low=$(echo
"scale=2;$thrs - $diff"|bc); fi
```

```
thrs_high=$(echo "scale=2;$thrs + $diff"|bc );
cat ./data/node_cpu.log >>./data/engine.log
echo thrs,mean,diff,min,max >> ./data/engine.log
echo $thrs,$mean,$diff,$min,$max >>./data/engine.log
echo -e "Middle Band \n" >>./data/engine.log
echo $thrs_low, $thrs_high >> ./data/engine.log
cat /dev/null > ./data/nodestatus
echo "done!"
echo "####### Separating Hosts based on load into High/Medium/Low groups #######"
while read line
        do
        node=`echo $line|awk -F, '{print $1}'`
        cpu=`echo $line|awk -F, '{print $2}'`
if [ $(echo "$cpu <= $thrs_low"|bc) -eq 1 ]; then echo $node,lowband
>>./data/nodestatus;fi
        if [ $(echo "$cpu <= $thrs_high"|bc) -eq 1 ] && [ $(echo "$cpu > $thrs_low"|bc) -
        eq 1 ]; then echo $node,midband >>./data/nodestatus;fi
        if [ $(echo "$cpu > $thrs_high"|bc) -eq 1 ]; then echo
        $node,highband>>./data/nodestatus;fi
done < ./data/node_cpu.log
echo "done!!!"
grep low ./data/nodestatus >/dev/null
chk1=$(echo $?)
```

```
grep high ./data/nodestatus >/dev/null
chk2=$(echo $?)
if [ $chk1 -ne 0 ] || [ $chk2 -ne 0 ]
        then
        echo "No High/Low Host Present. Quitting !"
        exit
fi
cat /dev/null >./data/nodes.low
cat /dev/null >./data/nodes.high
c=100
grep low ./data/nodestatus |awk -F, '{print $1}'>./data/nodes.low
for i in $(cat ./data/nodes.low)
do
        use=$(grep $i ./data/node_cpu.log|awk -F, '{print $2}')
if [ $(echo "$use < $c"|bc) -eq 1 ]
        then
        c=$use
fi
done
node_free=$(grep $c ./data/node_cpu.log|awk -F"," '{print $1}'|head -1)
work_trf=$(echo $thrs - $c + 10 |bc )
echo work_trf=$work_trf >>./data/engine.log
#### most free node selected
```

```
### now mixing all vms from high nodes
for hnodes in $(grep high ./data/nodestatus |awk -F, '{print $1}')
        do
        while read line
        do
        echo $hnodes,$line >>./data/nodes.high
        done < ./data/$hnodes.vm
        done
##done mixing highnodes/vmnamees and vmcpu
echo $node_free,$c
k=0
for i in $(cat ./data/nodes.high |awk -F, '{print $2}')
        do
        if [ $(echo "$i < $work_trf"|bc) -eq 1 ] && [ $(echo "$i > $k"|bc) -eq 1 ]
        then
        k=$i
        fi
done
if [ $(echo "$k != 0"|bc) -eq 1 ]
        then
        vm_mov=$(grep $k ./data/nodes.high|head -1 |awk -F"," '{print $3}')
        node_mov=$(grep $k ./data/nodes.high|head -1 |awk -F"," '{print $1}')
else
```

```
echo "No VM suitable for migration. Quitting !"
exit
fi
echo "VM = $vm_mov will be migrated from $node_mov to $node_free"
echo "VM running on Source Host : " $node_mov
ssh $node_mov virsh list
echo "VMs running on Destination Host : " $node_free
ssh $node_free virsh list
ssh $node_mov virsh migrate $vm_mov qemu+tcp://root@$node_free/system
if [ $? -eq 0 ]
        then
        echo "Migration Successful"
        echo "VMs running on Source Host :" $node_mov
        ssh $node_mov virsh list
        echo "VMs running on Destination Host : " $node_free
        ssh $node_free virsh list
fi
// Get data Codes
c=0
for k in `cat ./data/nodeinfo`
        do
        cat /dev/null > ./data/$k.vm
        ssh root@$k top -c -bn 1 |grep /usr/bin/kvm |grep -v "grep" |awk '{print
```

```
        $1,",",$9}' > ./data/tmp_pid
        ssh root@$k ps -ef|grep /usr/bin/kvm |grep -v "grep" |awk '{print $2,",",$18}' >
./data/tmp_vm
for i in `cat ./data/tmp_pid| awk '{print $1}'`
        do
        cpu=$(grep $i ./data/tmp_pid|awk -F, '{print $2}')
        vm=$(grep $i ./data/tmp_vm|awk -F, '{print $2}')
        echo $cpu, $vm >>./data/$k.vm
done
done
//VM Select Codes
c=100
grep low ./data/nodestatus |awk -F, '{print $1}'>./data/nodes.low
for i in $(cat ./data/nodes.low)
        do
        use=$(grep $i ./data/node_cpu.log|awk -F, '{print $2}')
        if [ $(echo "$use < $c"|bc) -eq 1 ]
        then
        c=$use
        fi
done
node_free=$(grep $c ./data/node_cpu.log|awk -F"," '{print $1}'|head -1)
        echo "Free'st node is :"  $node_free
```

```
#### most free node selected

### now mixing all vms from high nodes

for hnodes in $(grep high ./data/nodestatus |awk -F, '{print $1}')

        do

while read line

        do

        echo $hnodes,$line

        done < ./data/$hnodes.vm

        done

##done mixing highnodes/vmnamees and vmcpu
```

Database Part

Engine Logs

node1,83.38

node2,96.4

node3,3.08

thrs,mean,diff,min,max

60.95,49.74,11.21,3.08,96.4

Middle Band

49.74, 72.16

work_trf=67.87

// Nodes Details

node1,83.38

node2,96.4

node3,3.08

node1,highband

node2,highband

node3,lowband

8.2 GUI Development

GUI is developed in GtkPython Language. PyGTK is a module for the Python programming language which allows access to the GTK+ Graphical User Interface (GUI) toolkit. "GTK+ is a highly usable, feature rich toolkit for creating graphical user interfaces which boasts cross platform compatibility and an easy to use API." With the PyGTK binding, you can create advanced graphical layouts so that your users are able to interact easily with the application, and the API (the Application Programming Interface). Python is an extensible, object-oriented interpreted programming language which is provided with a rich set of modules providing access to a large number of operating system services, internet services (such as HTML, XML, FTP, etc.), graphics (including OpenGL, TK, etc.), string handling functions, mail services (IMAP, SMTP, POP3, etc.), multimedia (audio, JPEG) and cryptographic services. In addition there are many other modules available from third parties providing many other services.

// Important Codes for GUI

```
<?xml version="1.0" encoding="UTF-8"?>
<interface>
  <requires lib="gtk+" version="2.24"/>
  <!-- interface-naming-policy project-wide -->
  <object class="GtkWindow" id="window1">
    <property name="can_focus">False</property>
```

```
<child>
  <object class="GtkVBox" id="vbox1">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <child>
      <object class="GtkLabel" id="label1">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">KVM Load Balancer
Dashboard</property>
      </object>
      <packing>
        <property name="expand">False</property>
        <property name="fill">True</property>
        <property name="position">0</property>
      </packing>
    </child>
    <child>
      <object class="GtkHBox" id="hbox1">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <child>
          <object class="GtkLabel" id="label2">
```

```
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="label" translatable="yes">Use "Run" button to run the
balancer and "Refresh" to refresh the output below</property>
          </object>
          <packing>
            <property name="expand">True</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkButton" id="run">
            <property name="label" translatable="yes">Run</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <property name="xalign">0.54000002145767212</property>
            <property name="yalign">0.50999999046325684</property>
            <signal name="clicked" handler="on_run_clicked" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
```

```
      <property name="fill">True</property>
      <property name="position">1</property>
    </packing>
  </child>
  <child>
    <object class="GtkButton" id="refresh">
      <property name="label" translatable="yes">Refresh</property>
      <property name="visible">True</property>
      <property name="can_focus">True</property>
      <property name="receives_default">True</property>
      <signal name="clicked" handler="on_refresh_clicked" swapped="no"/>
    </object>
    <packing>
      <property name="expand">False</property>
      <property name="fill">True</property>
      <property name="position">2</property>
    </packing>
  </child>
</object>
<packing>
  <property name="expand">False</property>
  <property name="fill">True</property>
  <property name="position">1</property>
```

```
  </packing>
</child>
<child>
  <object class="GtkTextView" id="textview1">
    <property name="width_request">0</property>
    <property name="height_request">0</property>
    <property name="visible">True</property>
    <property name="can_focus">True</property>
  </object>
  <packing>
    <property name="expand">True</property>
    <property name="fill">True</property>
    <property name="position">2</property>
  </packing>
</child>
<child>
  <object class="GtkLabel" id="label3">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <property name="label" translatable="yes">Created </property>
  </object>
  <packing>
    <property name="expand">False</property>
```

```
            <property name="fill">True</property>
            <property name="position">3</property>
          </packing>
        </child>
      </object>
    </child>
  </object>
</interface>
// Calling File
import gtk
import gobject
import pango
import os
from subprocess import Popen, PIPE
import fcntl
wnd = gtk.Window()
wnd.set_default_size(800, 600)
wnd.connect("destroy", gtk.main_quit)
wnd.set_title("KVM Load Balancer")
textview = gtk.TextView()
fontdesc = pango.FontDescription("monospace")
textview.modify_font(fontdesc)
scroll = gtk.ScrolledWindow()
```

```
scroll.add(textview)
exp = gtk.Expander("Balancer is now running . Click Icon to see log below")
exp.add(scroll)
wnd.add(exp)
wnd.show_all()
sub_proc = Popen("kvmbalance", stdout=PIPE, shell=True)
sub_outp = ""
def non_block_read(output):
    fd = output.fileno()
    fl = fcntl.fcntl(fd, fcntl.F_GETFL)
    fcntl.fcntl(fd, fcntl.F_SETFL, fl | os.O_NONBLOCK)
    try:
        return output.read()
    except:
        return ''
def update_terminal():
    textview.get_buffer().insert_at_cursor(non_block_read(sub_proc.stdout))
    return sub_proc.poll() is None
gobject.timeout_add(100, update_terminal)
gtk.main()
```

CHAPTER IX
Performance and Result Analysis

In order to assess the performance of the algorithm, a prototype system of VM management was developed. Virtual platform: KVM and storage system: NFS was used. A physical machine was chosen as the host machine. QEMUKVM and Virtual Machine Manager were installed to manage and schedule VM; and its operating system was CentOS7, CPU: Intel Core i5-2400 3.10 GHz * 4, and Memory: 3 GB. Three client machines of same configuration as above were chosen. libvirt, qemu-kvm, virt-manager, nfs server packages were installed.

Test Case 1:-

In this scenario the CPU usage sent to policy engine is the average of CPU usage over last three minutes of respective hosts. The policy engine calculates thresholds every five minutes and takes the decision for load balancing i.e. whether to migrate some VM or not. The experimental results are as follows (CPU USAGE in percentage).Figure 7 to 17 depicts the hosts and their corresponding CPU usage after three iterations of the algorithm. Followed by Fig. 18 which depicts the variation in CPU usage of hosts over period of 25 minutes. The usage of CPU by individual VMs was approximately constant over 25 minutes and each VM was running on single core. By using the formulae given in Load Evaluation, lower and upper limit of moderate band are calculated. Hence Host 1 is in moderately loaded band, Host 2 is in lightly loaded band and Host 3 in heavily loaded band and A2 is migrated from Host 3 to Host 2. (Figure 9) By using the formulae given in Load Evaluation, lower and upper limit of moderate band are calculated. Hence Host 1 is in moderately loaded band, Host 3 is in lightly loaded band and Host 2 is in HLB and K2 is migrated from Host 2 to Host 3. (Figure 12).

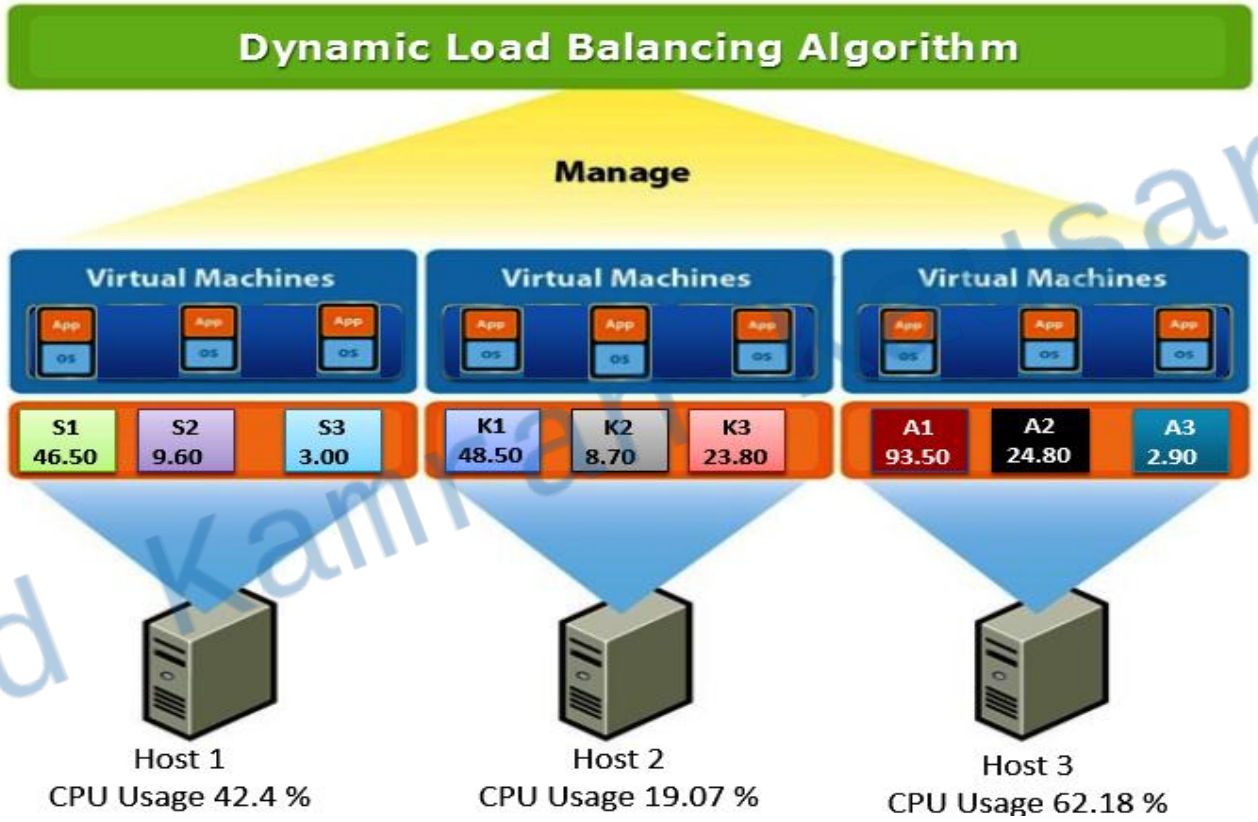


Figure7: Snapshot of Hosts with VMs before Migration

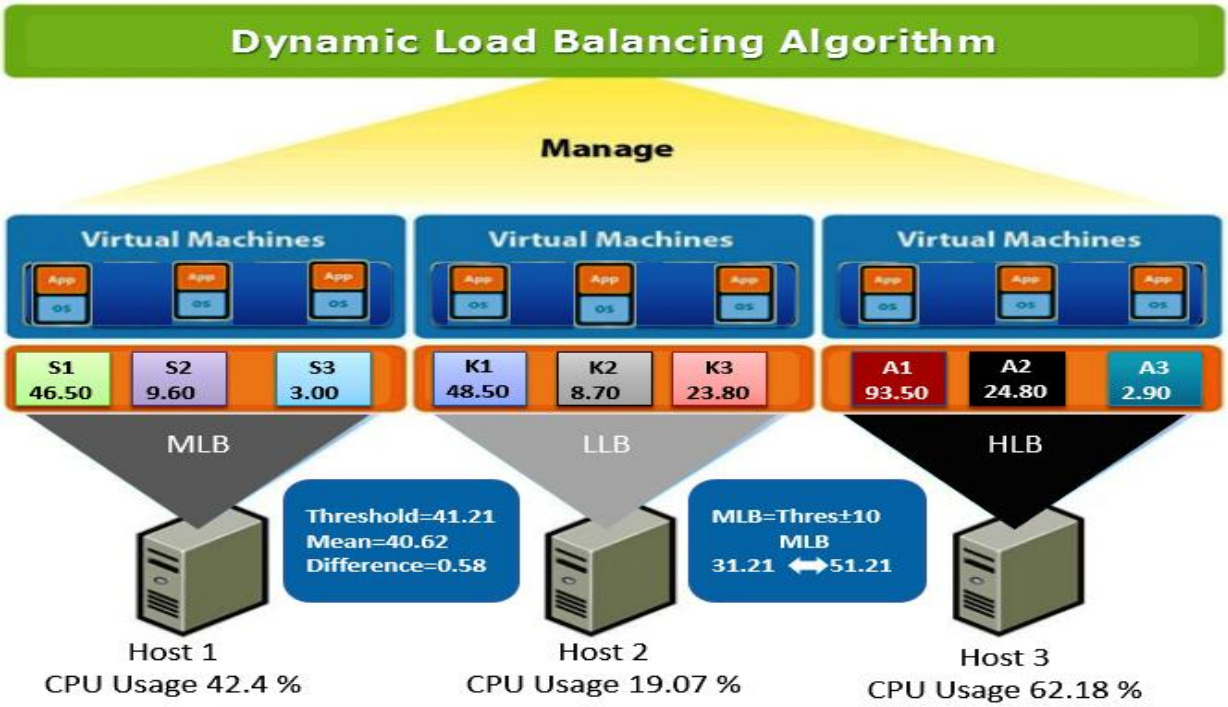


Figure8: Snapshot of Hosts with VMs before Migration and allocation of bands to Hosts.

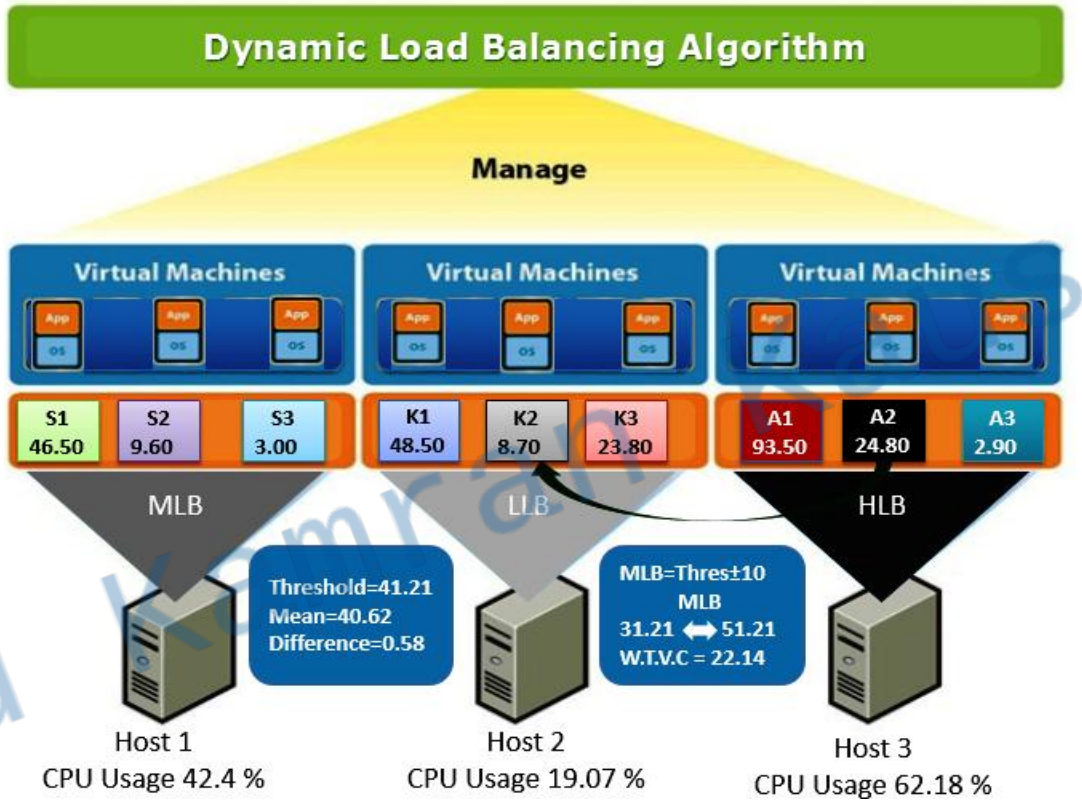


Figure9:In 1st Iterations of Algorithm.VM (A2) moves from Host3 (HLB) to Host2(LLB)

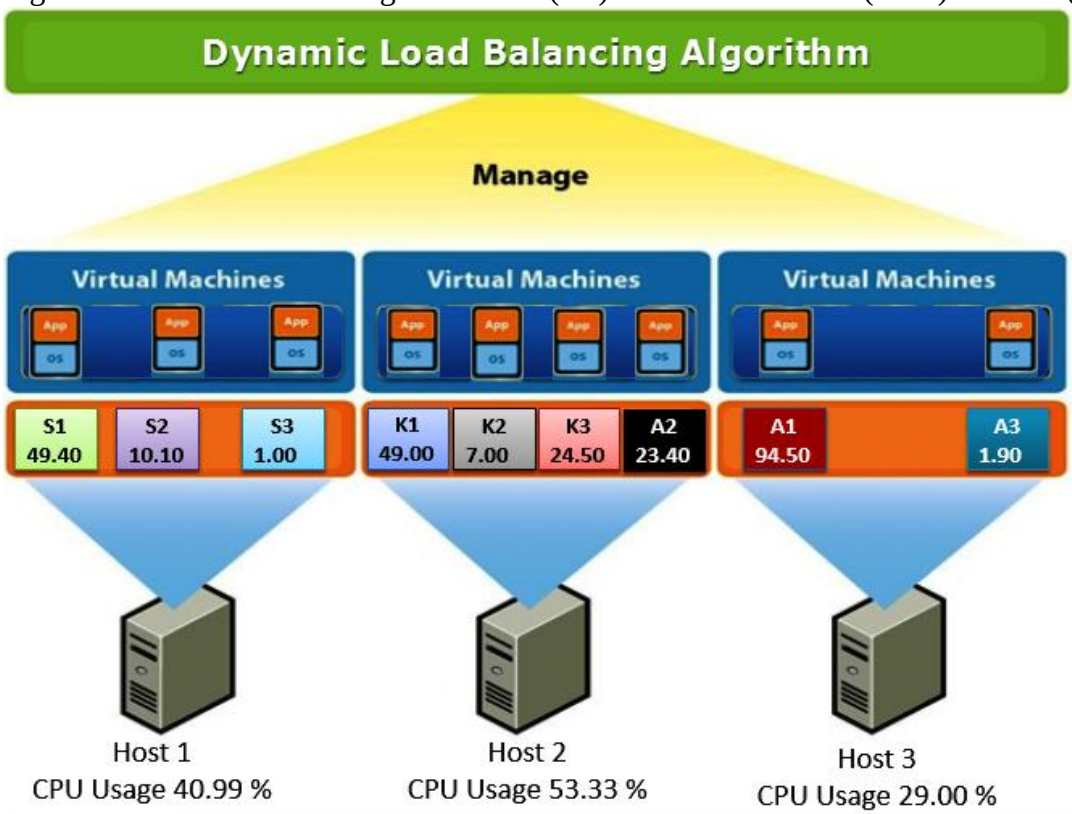


Figure10: Snapshot of Hosts with VMs after 1st Migration.

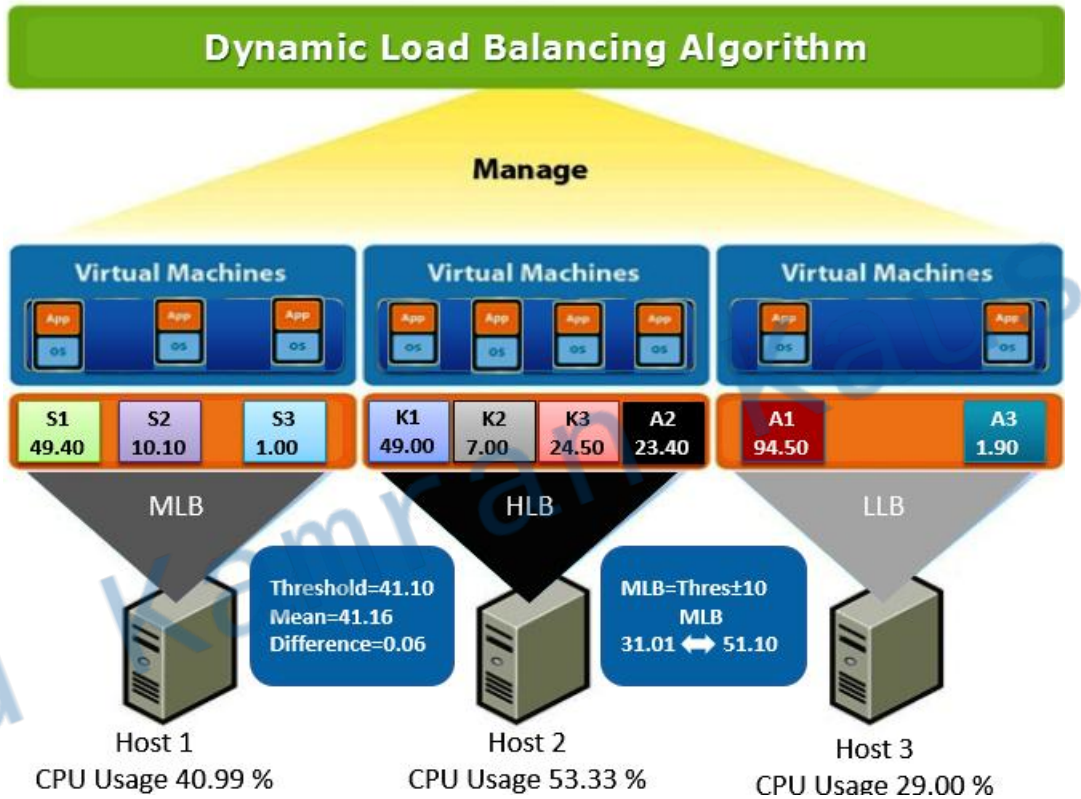


Figure11: Snapshot of Hosts with VMs after 1st Migration and bands allocated to Hosts.

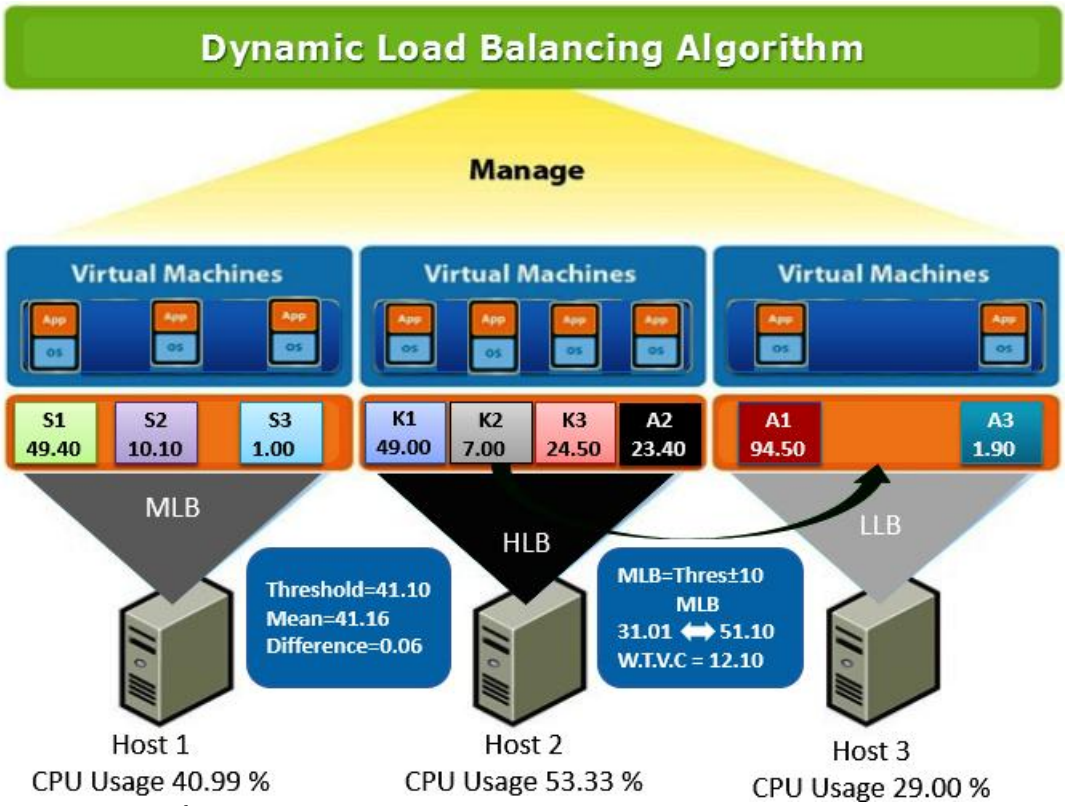


Figure12:In 2nd Iterations of Algorithm.VM(K2) moves from Host2(HLB) to Host3(LLB)

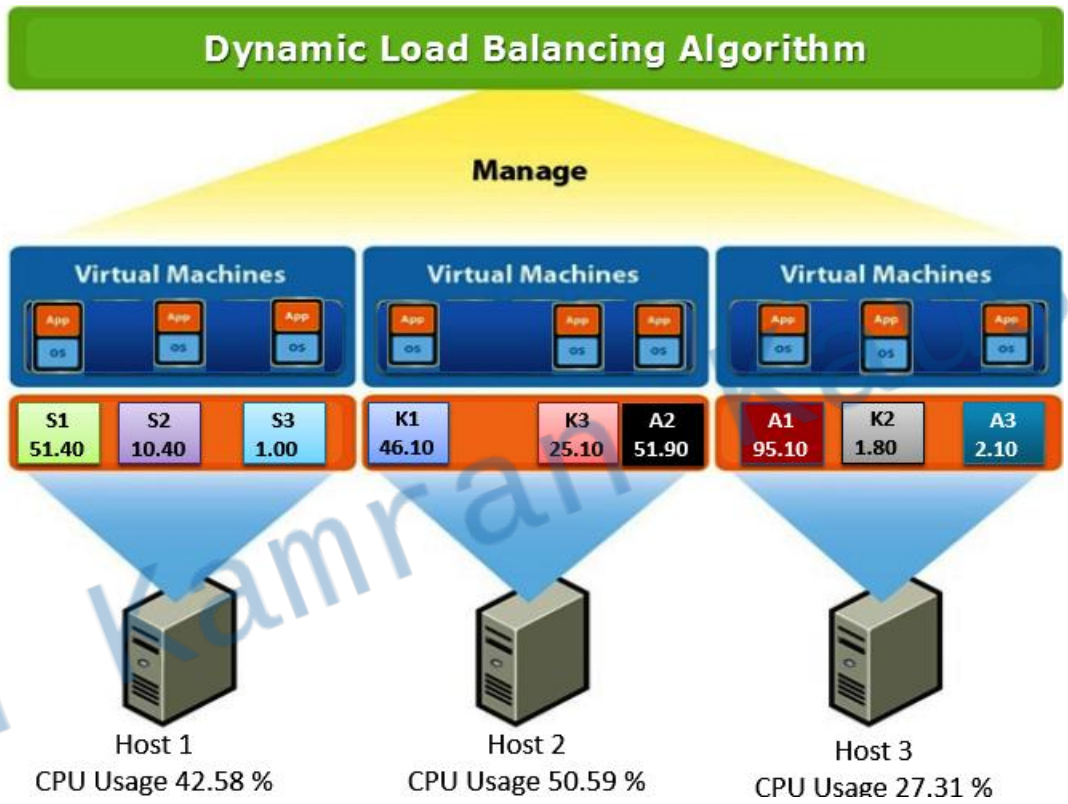


Figure13: Snapshot of Hosts with VMs after 2nd Migration.

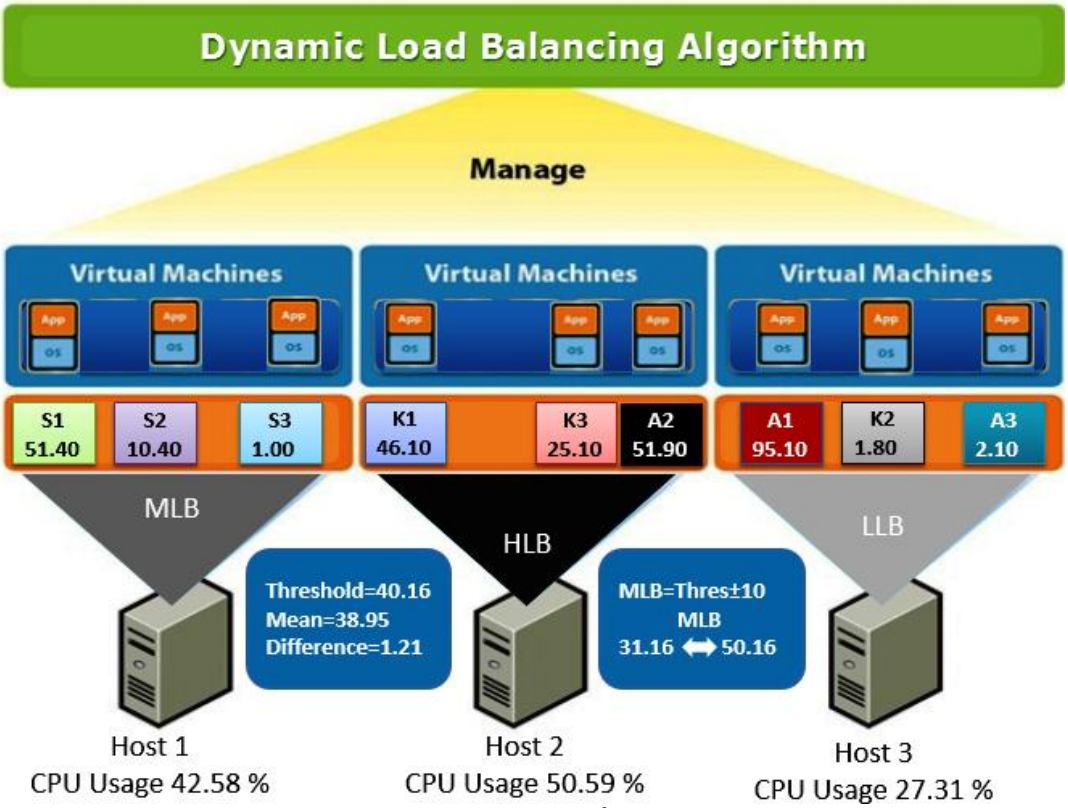


Figure14: Snapshot of Hosts with VMs after 2nd Migration and bands allocated to Hosts.

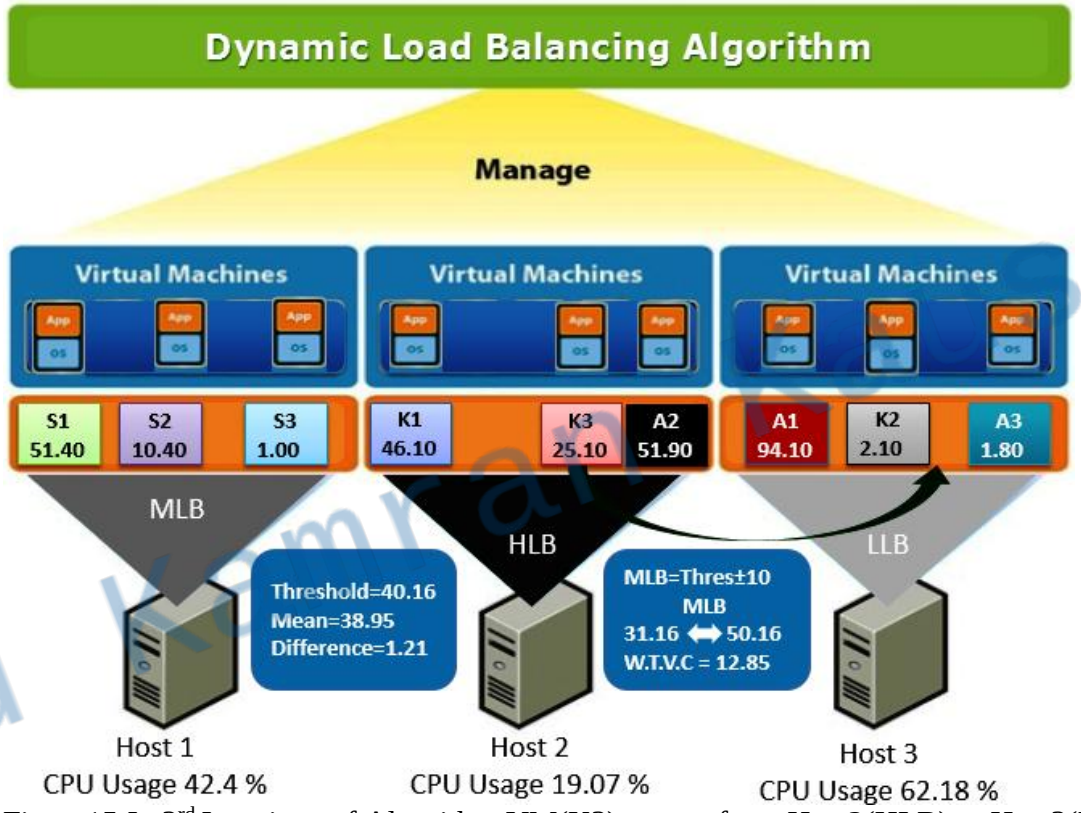


Figure15:In 3rd Iterations of Algorithm.VM(K3) moves from Host2(HLB) to Host3(LLB)

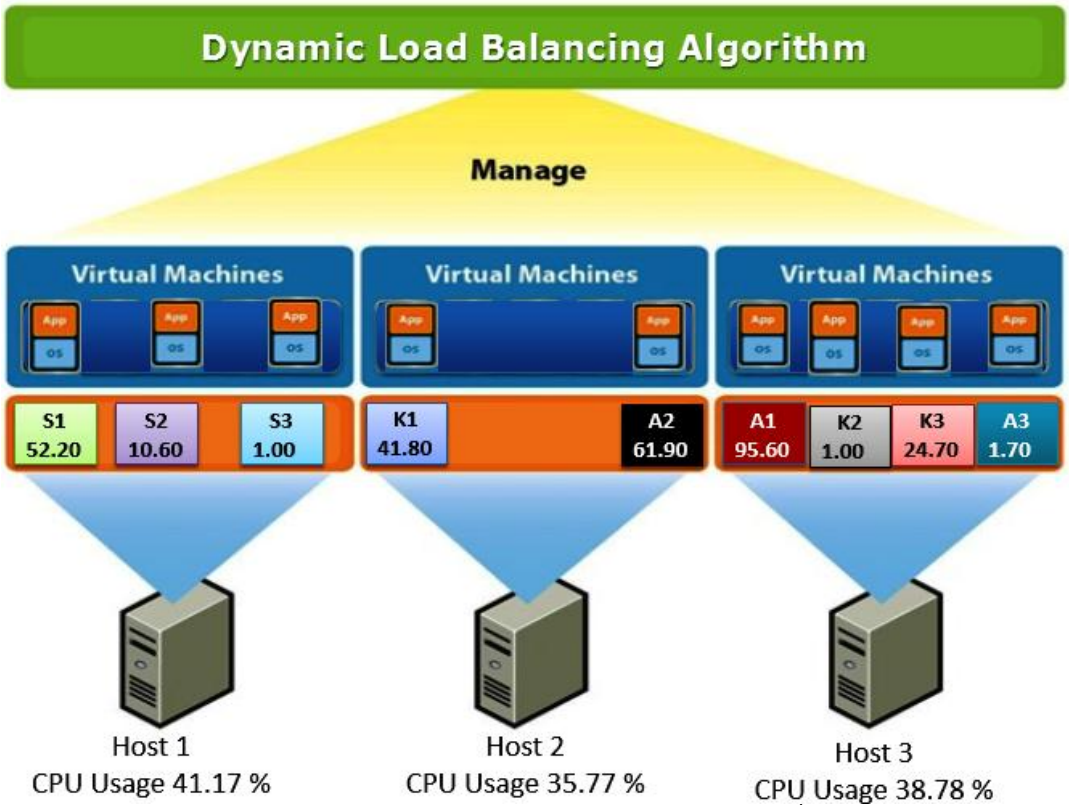


Figure16: Snapshot of Hosts with VMs after 3rd Migration.

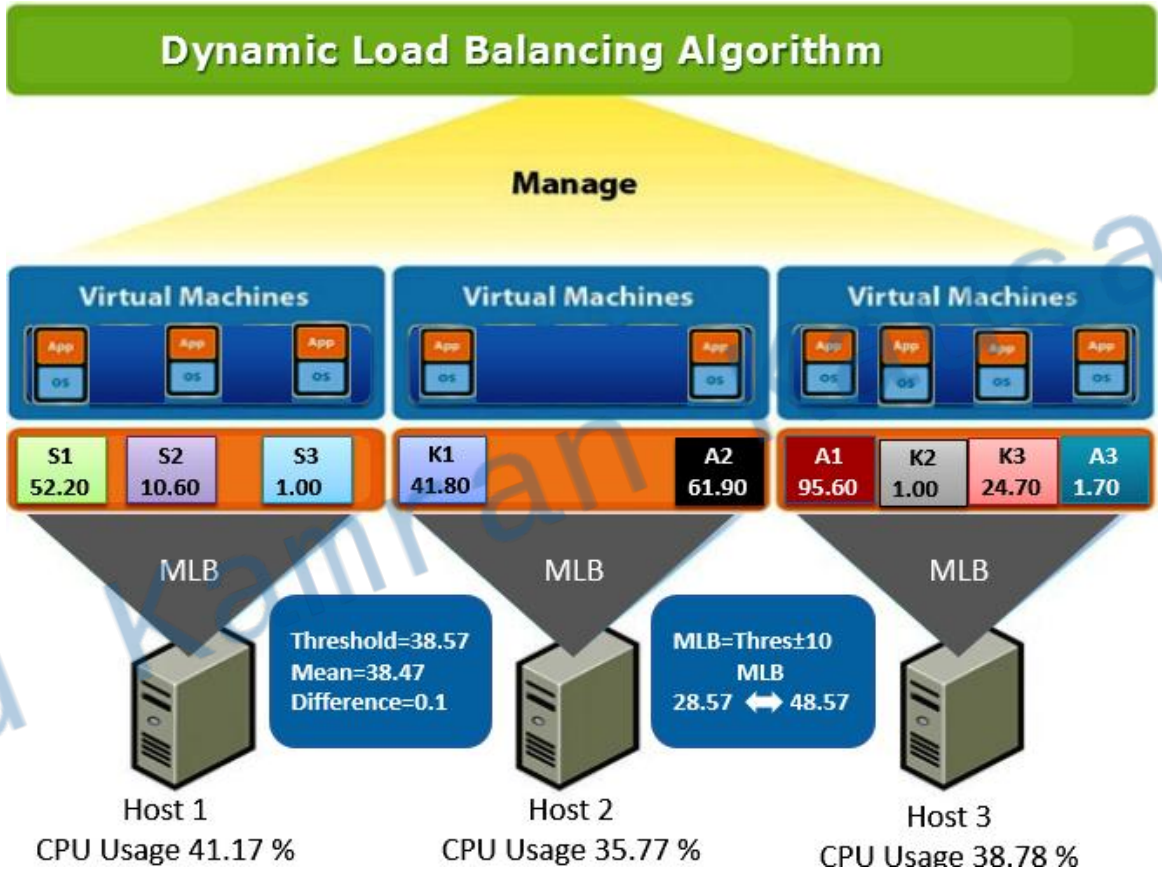


Figure17: Snapshot of Hosts with VMs, all Hosts are in MLB no migration needed.

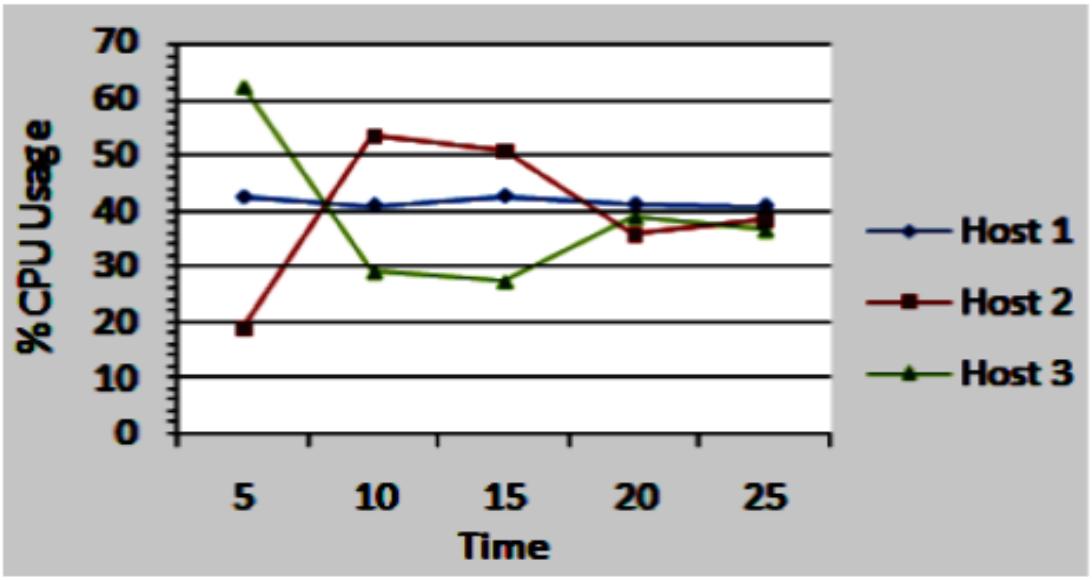
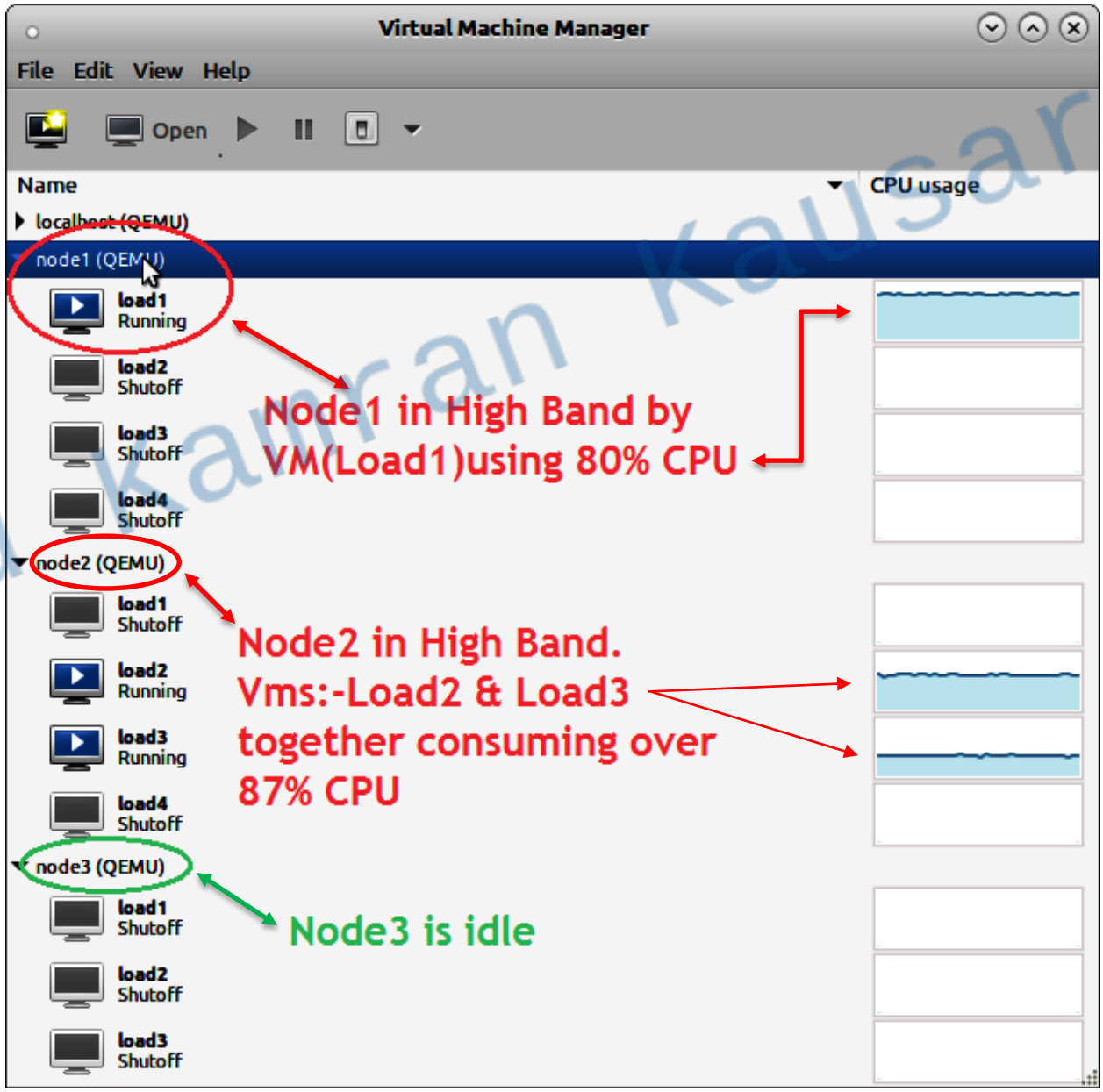
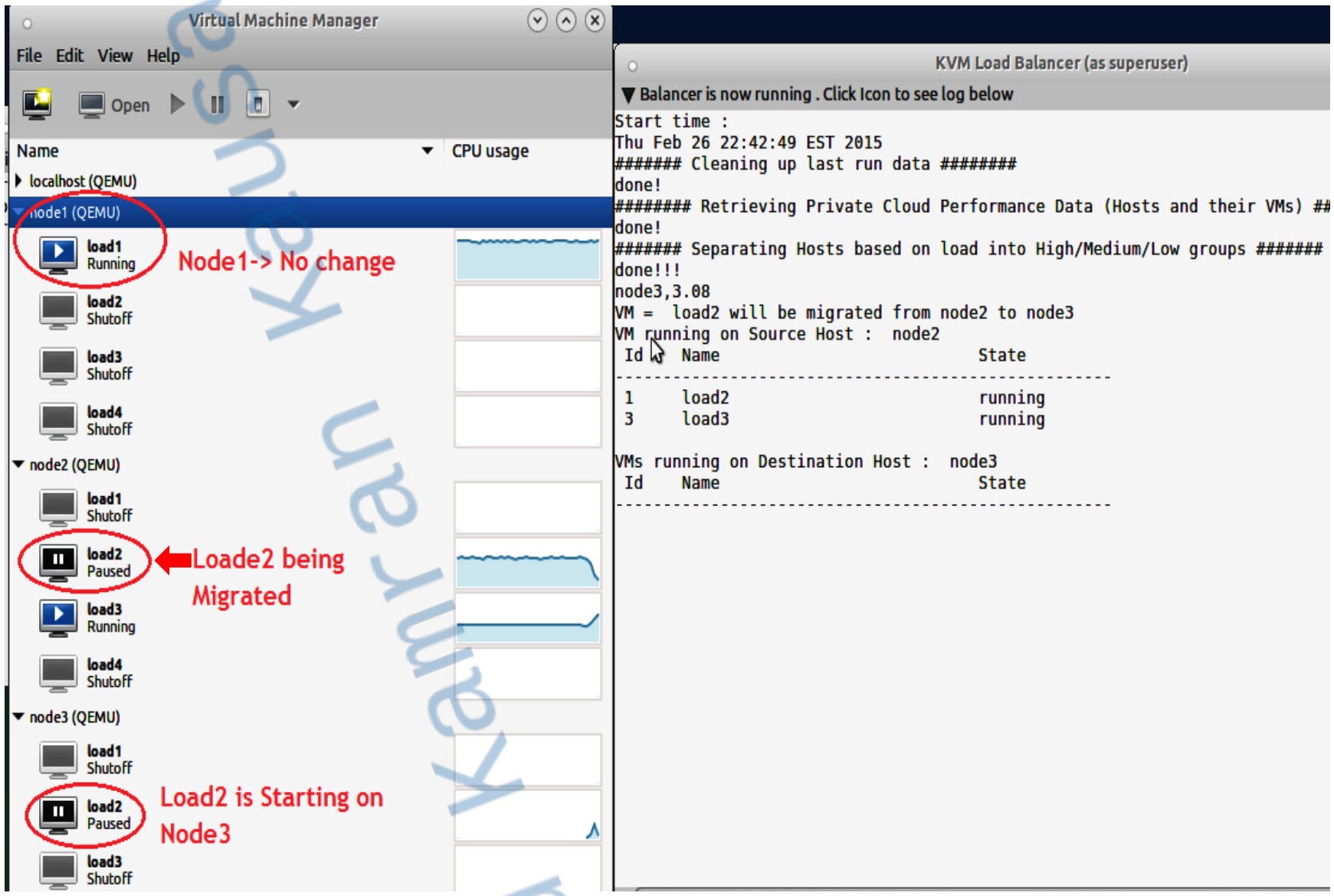
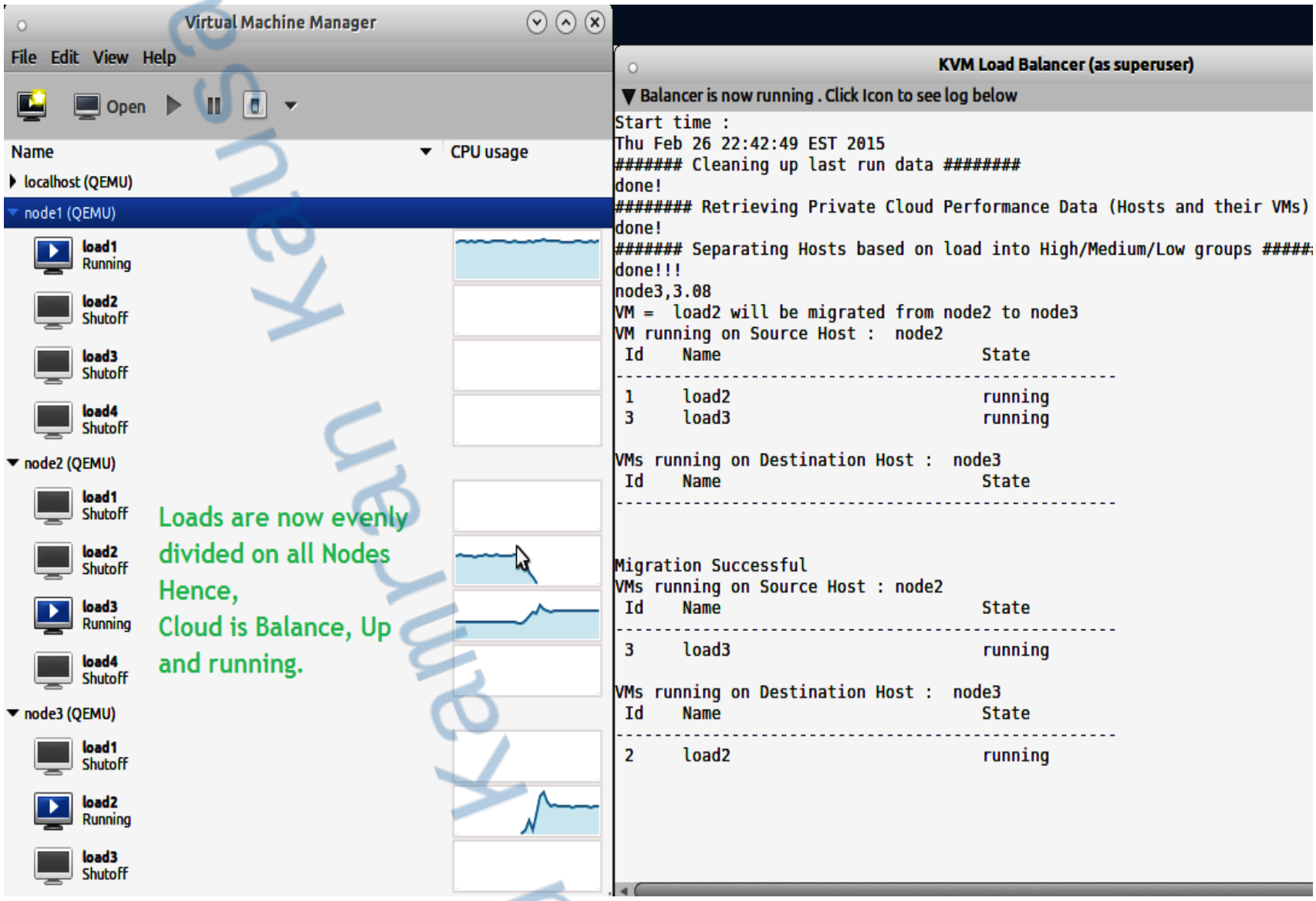


Figure18: Variation in CPU usage of hosts over period of 25 minutes.

Test Case 2:-







Comparison of achieved (Modified) Algorithm and Original Algorithm

Modified Algorithm	Original Algorithm
1. Has only Four Phases. Making Algorithm simple and efficient.	**1.Has Five Phases**
2. In this Algorithm fewer nodes fall in Higher band and Low band.	**2. Greater numbers of nodes will fall in High band and Low Band complicating VMs Selection.**
3.Works in all Scenario Because Modified Algo Checks all nodes in High band and Low band for best possible migration. **Consider the scenario describe below** **Node1:-** **VM1=80% of CPU** **Node2:-** **VM2=60% of CPU** **VM3=60% of CPU** **Node3:- Idle** **Outcome:-** ***VM2 (60% of CPU) is migrated from Node2 to Node3.***	**3.Fails in certain Scenario Because tries to migrate VM from first node in High band to First node in Low Band.** **Consider the scenario describe below** **Node1:-** **VM1=80% of CPU** **Node2:-** **VM2=60% of CPU** **VM3=60% of CPU** **Node3:- Idle** **Outcome:-** ***No Migration occurs.***
4.Transfer max work possible between nodes in each migration **Consider the scenario describe below** **Node1:-** **VM1=40% of CPU** **VM2=40% of CPU** **Node2:-** **VM3=40% of CPU** **VM3=50% of CPU**	**4. Transfer just the first VM suitable for migration. Does not compare with other VMs** **Consider the scenario describe below** **Node1:-** **VM1=40% of CPU** **VM2=40% of CPU** **Node2:-** **VM3=40% of CPU** **VM3=50% of CPU**

Node3:- Idle **Outcome:-** ***VM4 (50% of CPU) is migrated from Node2 to Node3.***	**Node3:- Idle** **Outcome:-** ***VM1 (40% of CPU) is migrated from Node1 to Node3.***

Table2: Comparison of Algorithms

CHAPTER X
Conclusion and Future Work

The work has proposed a policy engine to dynamically balance the load over the network. Originally the network was imbalanced. There were hosts in heavily as well as lightly loaded bands. After some iterations (three in the above scenario) of the load balancing algorithm, all the hosts were balanced i.e. all the hosts were in the moderately loaded band (Fig. 18).

Cloud Computing is a vast area and load balancing plays a very important role in case of Cloud. The work has focused on CPU usage as load parameter that is applied to the setup, but there are still other parameters and approaches that can be applied to balance the load. The performance of the given algorithm can be increased by varying different parameters like memory usage, disk I/O, network load. Further dynamic load balancing can be improved and the Live Migration algorithm implemented in QEMU-KVM can be optimized, so that migration time will be reduced and performance will also be improved.

REFERENCES

[1] Jyotiprakash Sahoo, Subasish Mohapatra, Radha Lath, Virtualization: A Survey On Concepts, Taxonomy And Associated Security Issues, Second International Conference on Computer and Network Technology, 2010.

[2] Ali M. Alakeel, A Guide to Dynamic Load Balancing in Distributed Computer Systems, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.

[3] Terry C. Wilcox Jr, Dynamic Load Balancing Of Virtual Machines Hosted On Xen, Department of Computer Science, Brigham Young University, M.S. Thesis, April 2009.

[4] Youran Lan, Ting Yu, A Dynamic Central Scheduler Load Balancing Mechanism, Computers and Communications, pp 734-740, May 1995.

[5] Yi Zhao, Wenlong Huang, Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud, Fifth International Joint Conference on INC, IMS and IDC, 2009.

[6] F. Ma, F. Liu, Z. Liu, Live Virtual Machine Migration Based on Improved Pre-copy Approach, Proc. Software Engineering and Service Sciences, pp. 230-233, 2010.

[7] Geoffroy Vallee, Thomas Naughton, Christian Engelmann, Stephen L Scott, Hong Ong, System-level Virtualization For High Performance Computing, 2008.

[8] Jerrell Watts, Stephen Taylor, A Practical Approach to Dynamic Load Balancing, IEEE Transactions on Parallel and Distributed Systems, Feb 1998.

[9] KVM Kernel Based Virtual Machine Red Hat, Inc. 2009.

[10] Christopher Clark et. al., Live Migration of Virtual Machines, 2nd Symposium on Networked Systems Design & Implementation, NSDI'05

[11] WWW.WIKIPEDIA.ORG

[12] WWW.PYTHON.ORG

[13] Lee, R. & Jeng, B. (2011). Load-balancing tactics in cloud. In proc. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery CyberC), IEEE, (pp. 447-454).

[14] Radojevic, B. & Zagar, M. (2011). Analysis of issues with load balancing algorithms in hosted (cloud) environments. In proceedings of 34th International Convention on MIPRO, IEEE.

[15] Luo, S., Lin, Z. & Chenm, X. (2011). Virtualization security for cloud computing service. 2011 International Conference on Cloud and Service Computing 978-1-4577-1637-9/11/$26.00 ©2011 IEEE. Shenzhen, China: ZTE Corporation.

[16] Al Nuaimi, K., Mohamed, N., Al Nuaimi, M. & Al-Jaroodi, J. (2012). A survey of load balancing in cloud computing: challenges and algorithms.2012 IEEE Second Symposium on Network Cloud Computing and Applications 978-0-7695-4943-9/12 $26.00 © 2012 IEEE DOI 10.1109/NCCA.2012.29.College of Information Technology, UAEU Al Ain, United Arab Emirates

[17] Ms.NITIKA "Comparative Analysis of Load Balancing Algorithms in Cloud Computing" International Journal of Engineering and Science Vol 1-Issue 1.

[18] Kousik Dasguptaa, Brototi Mandalb, Paramartha Duttac, Jyotsna Kumar Mondald, Santanu Dame, A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing" International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA) vol 10,2013.

[19] Shilpa V Pius, Shilpa T S "Survey on Load Balancing in Cloud Computing"

International Conference on Computing,Communication and Energy Systems (ICCCES-2014).

[20] Mohiuddin Ahmed, Abu Sina Md. Raju Chowdhury, Mustaq Ahmed, Md. Mahmudul Hasan Rafee "An Advanced Survey on Cloud Computing and State-of-the-art Research Issues" IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012.