SRI SAI RAM ENGINEERING COLLEGE

SAILEO NAGAR, WEST TAMBARAM,

CHENNAI-44

(Accredited and ISO Certificate Institution)

PROSPERITY THROUGH TECHNOLOGY

LAB MANUAL

IT8711 FOSS AND CLOUD COMPUTING LABORATORY

IV YEAR/ VII SEMESTER

DEPARTMENT OF INFORMATION TECHNOLOGY

IT8711 FOSS AND CLOUD COMPUTING LABORATORY

LIST OF EXPERIMENTS

1. Use gcc to compile c-programs. Split the programs to different modules and create an application using make command.
2. Use version control systems command to clone, commit, push, fetch, pull, checkout, reset, and delete repositories.
3. Install Virtualbox/VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8.
4. Install a C compiler in the virtual machine created using virtual box and execute Simple Programs
5. Install Google App Engine. Create *hello world* app and other simple web applications using python/java.
6. Use GAE launcher to launch the web applications.
7. Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.
8. Find a procedure to transfer the files from one virtual machine to another virtual machine.
9. Find a procedure to launch virtual machine using trystack (Online Openstack Demo Version)
10. Install Hadoop single node cluster and run simple applications like word count.

TABLE OF CONTENETS

STUDY OF CLOUDSIM

AIM:

To study about cloudsim and simulation of cloudsim with eclipse.

INTRODUCTION:

CloudSim is a simulation tool that allows cloud developers to test the performance of their provisioning policies in a repeatable and controllable environment, free of cost It is a simulator; hence, it doesn't run any actual software. It can be defined as 'running a model of an environment in a model of hardware', where technology-specific details are abstracted.CloudSim provides a generalised and extensible simulation framework that enables seamless modelling and simulation of app performance. By using CloudSim, developers can focus on specific systems design issues that they want to investigate, without getting concerned about details related to cloud-based infrastructures and services.

CloudSim is a library for the simulation of cloud scenarios. It provides essential classes for describing data centres, computational resources, virtual machines, applications, users, and policies for the management of various parts of the system such as scheduling and provisioning.It is flexible enough to be used as a library that allows you to add a desired scenario by writing a Java program.By using CloudSim, organisations, R&D centres and industry-based developers can test the performance of a newly developed application in a controlled and easy to set-up environment.

ARCHITECTURE OF CLOUDSIM:

The CloudSim layer provides support for modelling and simulation of cloud environments including dedicated management interfaces for memory, storage, bandwidth and VMs. It also provisions hosts to VMs, application execution management and dynamic system state monitoring. A cloud service provider can implement customised strategies at this layer to study the efficiency of different policies in VM provisioning.

The user code layer exposes basic entities such as the number of machines, their specifications, etc, as well as applications, VMs, number of users, application types and scheduling policies.

The main components of the CloudSim framework are

Regions: It models geographical regions in which cloud service providers allocate resources to their customers. In cloud analysis, there are six regions that correspond to six continents in the world.

Data centres: It models the infrastructure services provided by various cloud service providers. It encapsulates a set of computing hosts or servers that are either heterogeneous or homogeneous in nature, based on their hardware configurations.

Data centre characteristics: It models information regarding data centre resource configurations.

Hosts: It models physical resources (compute or storage).

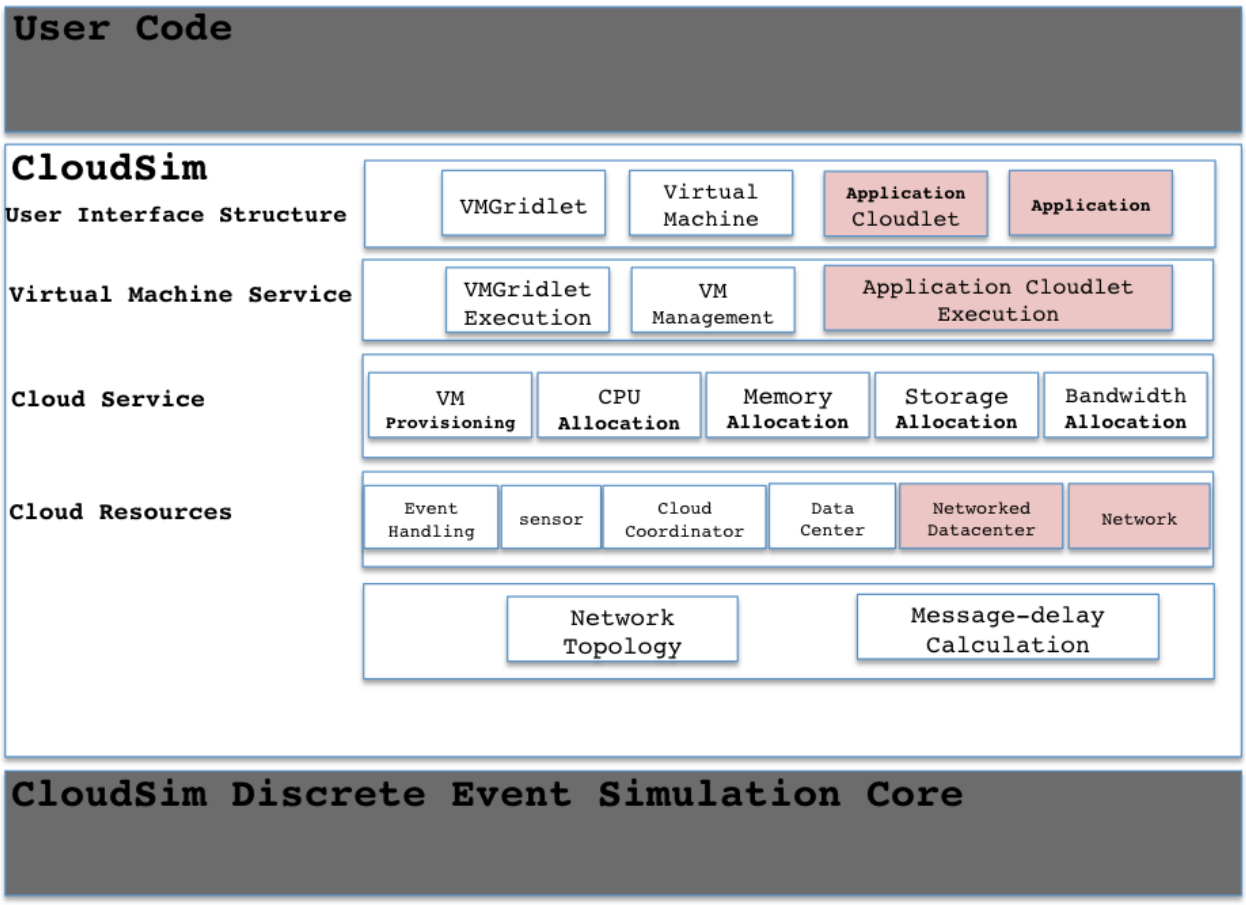
The user base: It models a group of users considered as a single unit in the simulation, and its main responsibility is to generate traffic for the simulation.

Cloudlet: It specifies the set of user requests. It contains the application ID, name of the user base that is the originator to which the responses have to be routed back, as well as the size of the request execution commands, and input and output files. It models the cloud-based application services. CloudSim categorises the complexity of an application in terms of its computational requirements. Each application service has a pre-assigned instruction length and data transfer overhead that it needs to carry out during its life cycle.

Service broker: The service broker decides which data centre should be selected to provide the services to the requests from the user base.

VMM allocation policy: It models provisioning policies on how to allocate VMs to hosts.

VM scheduler: It models the time or space shared, scheduling a policy to allocate processor cores to VMs.



SIMULATION OF CLOUDSIM WITH ECLIPSE:

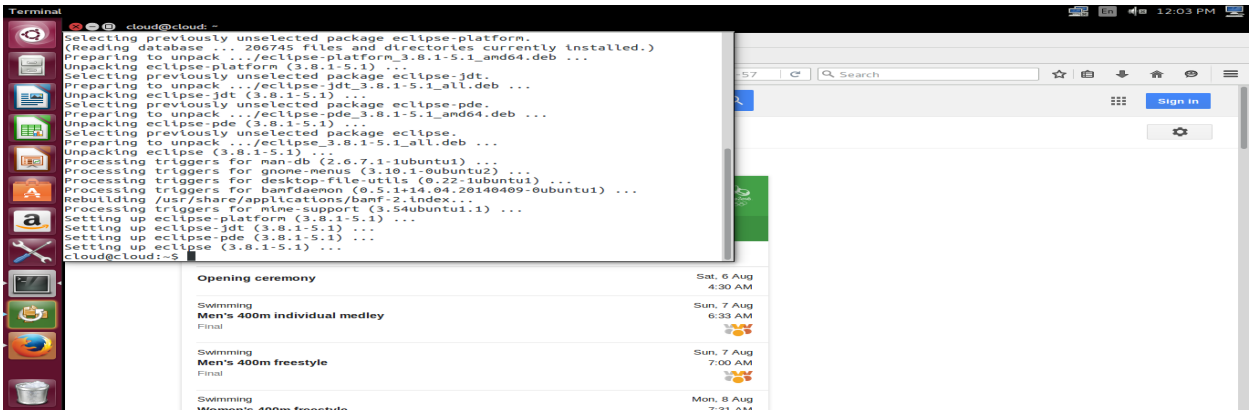
Prerequisites to install cloudsim

1) Install Java 1.7 or latest

sudo apt-get install openjdk-7-jdk

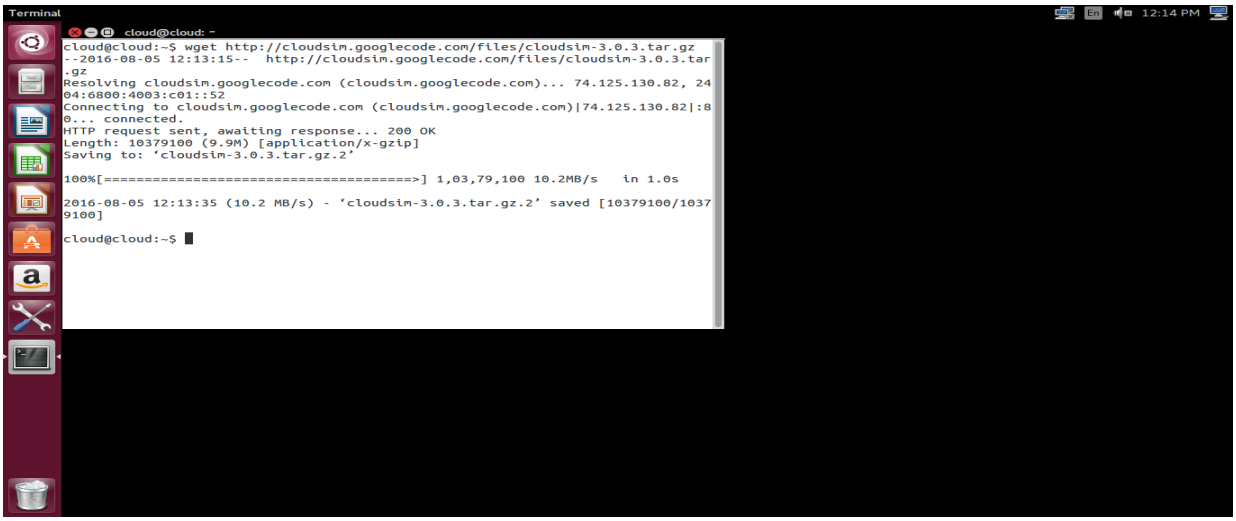
2) Installing Eclipse

sudo apt-get install eclipse



3) Download the latest cloudsim using below link

wget http://cloudsim.googlecode.com/files/cloudsim-3.0.3.tar.gz



4) Untar the cloudsim package using the following command

$tar zxvf cloudsim-3.0.3.tar.gz

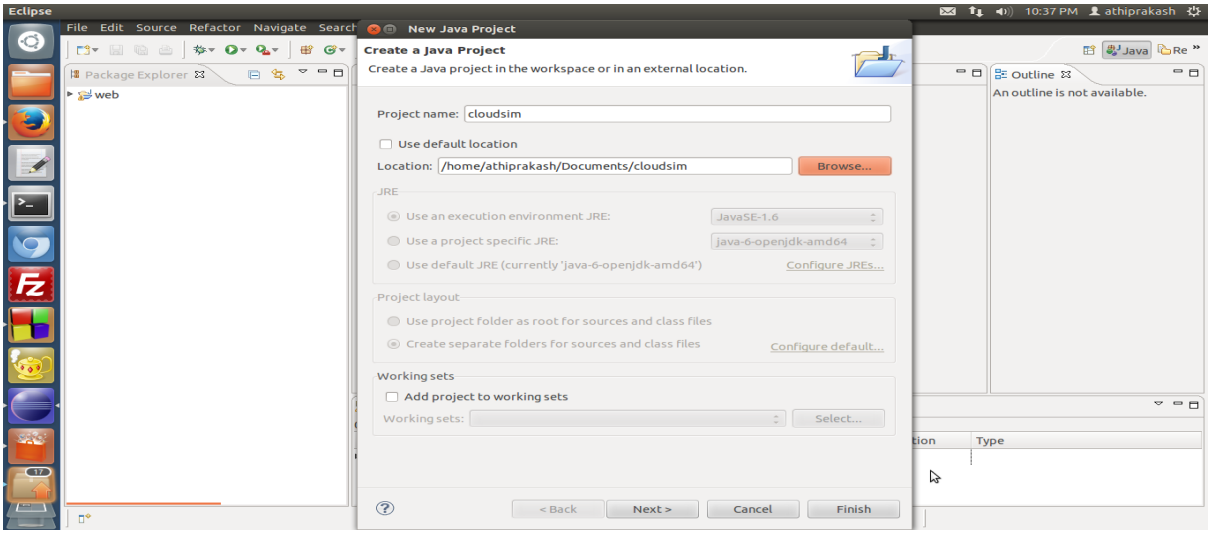
5) Go-to eclipse and create a workbench.

6) Now,create a new Java Project

To create a new project do the following,

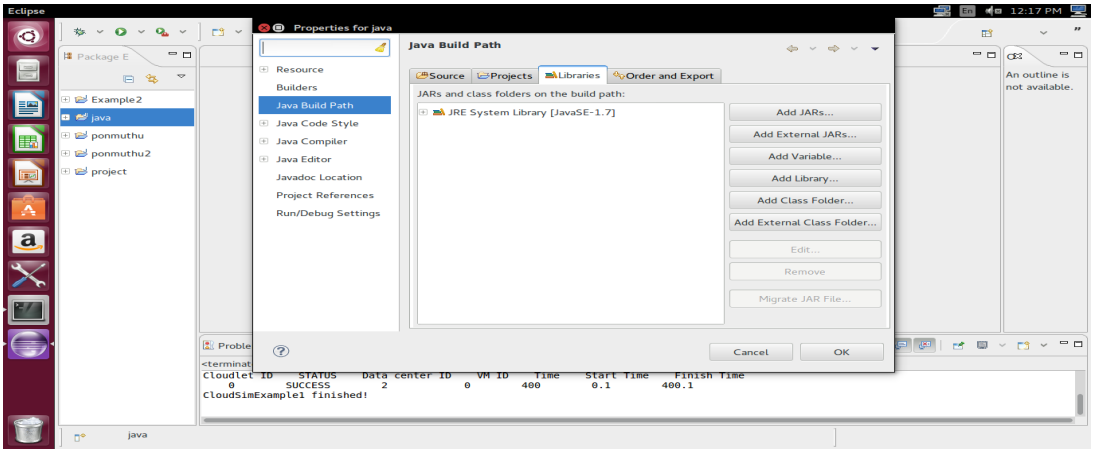
i) **eclipse=>File=> New=>Java Project**

ii) Give the name of the project and click finish



iii) Right click on the project and select **build path=>configure build path.**

iv) Choose the **libraries** tab and select **Add External Jars** and click **OK.**



v) click cloud and select **cloudsim-3.0.3.tar.gz** and click **OK.**

vi) Run an Example program to feel the cloudsim Simulation.

RESULT:

Thus the cloudsim and simulation of cloudsim with eclipse is studied.

CLOUD SIM WITH ONE DATACENTER

AIM:

To Create a data center with one host and one cloudlet in it using cloud sim program.

ALGORITHM:

1. Start
2. Initiating the cloud sim
3. Create a data center
   a) Create a list of type pelist and hostlist
   b) Add pelist to hostlist
   c) Create a javalist of storage
   d) Finally add the pelist,hostlist and storagelist into Datacentercharacteristics
   e) Create Datacenter with vmallocation policy
4. Create Datacenter broker
5. Create VM list
   a) Add to vm list
   b) Add the vm list to data center broker
6. Create a cloudlet
7. Start stimulation
8. Stop stimulation
9. End

CLOUD SIM WITH TWO DATACENTERS

AIM:

To create a two data centers with one host and two cloudlets using cloud sim program.

ALGORITHM:

1. Start
2. Initiating the cloudsim
3. Create a data center
   a) Create a list of type pelist and hostlist
   b) Add pelist to hostlist
   c) Create a javalist of storage
   d) Finally add the pelist,hostlist and storagelist into Datacentercharacteristics
   e) Create Datacenter with vmallocation policy
4. Create Datacenter broker
5. Create the two VM lists
   a) Add to vm list
   b) Add the vm list to data center broker
6. Create a two cloudlet lists
7. Start stimulation
8. Stop stimulation
9. Print the result about the cloudlets
10. End

CALCULATOR WEB SERVICE

AIM:

To create a calculator with basic arithmetic operations using EJB module and web service.

SERVER SIDE PROCEDURE:

1. Create new project
    a. New-> Project -> JavaEE -> EJBModule -> Next -> Give Project Name -> Finish.
2. Create new Web service
    a. Right click the created project name -> New -> Web Service -> Give name for Web service and Package -> Finish
    b. A new folder named "Web Services" will be created under the project folder. Inside that, new web services will be created.
3. Creating functions for web service
    a. Under the web services folder, right click the web service created -> Add Operation
        i. Give appropriate name for the operation.
        ii. Select the return type of the function.
        iii. Adding parameters :
            1. Click Add -> give name of the parameter and their data type.
            2. Follow the above step to add more parameters.
        iv. Click ok.
4. Write the required operation/statements inside the function and also see to that the return type matches. Save the file.
5. Deploying and testing :
    a. Right click the project -> Deploy
    b. In the web services folder, right click on the web service -> Test Web service.

CLIENT SIDE PROCEDURE:

1. Create a new project
    a. File -> New Project -> JavaWeb -> We Application ->Next -> Give a name -> Finish.
2. A file names index.jsp will be created. Modify the code as below:

```
<body>
    <label>a</label><input type="text" name="n1"/></br>
```

```
<label>b</label><input type="text" name="n2"/></br>
<form name="form1" action="add.jsp" method="get">
    <input type="submit" value="Add"/>
</form>
<form name="form2" action="sub.jsp" method="get">
    <input type="submit" value="Sub"/>
</form>
<form name="form3" action="mul.jsp" method="get">
    <input type="submit" value="Multiply"/>
</form>
<form name="form4" action="div.jsp" method="get">
    <input type="submit" value="Division"/>
</form>
</body>
```

3. Right click the created project -> New -> JSP -> Give name (add.jsp) -> Finish.
4. Create web service client.
    a. Right click the created project -> New -> Web Service Client.
    b. Enter the WSDL URL that can get form the following steps:
        i. In the firstly created EJB Module project, right click the Web Service under the Web Services folder -> properties.
        ii. A URL will be displayed. Copy it and paste it in the WSDL URL field.
    c. Click Finish.
5. Right click the payroll.jsp file in the editor -> Web Service Client Resources -> Call Web Service Operation. In the dialog box that opens, click the '+' sign until to get the required operation -> click Ok. The code for generating Web Service operation will be generated.
6. Modify the JSP file.
7. Right click the Web Application project -> Deploy.
8. Right click the Web Application project -> Run.
9. Index.jsp file will be displayed. Give appropriate inputs and click "Submit".
10. Result will be displayed.

Add.jsp

```
 <body>
<%! double addres,a,b;%>
```

```
<%
try {
  pack.CalcWebServiceService service = new pack.CalcWebServiceService();
  pack.CalcWebService port = service.getCalcWebServicePort();
   // TODO initialize WS operation arguments here
  a = Double.parseDouble(request.getParameter("n1"));
  b = Double.parseDouble(request.getParameter("n2"));
  // TODO process result here
  addres = port.add(a, b);

} catch (Exception ex) {
  // TODO handle custom exceptions here
}
%>
Number 1<input type="text" value="<%=a%>"><br/>
Number 2<input type="text" value="<%=b%>"><br/>
Added Value<input type="text" value="<%=addres%>">
</body>
```

OpenNebula - Private Cloud Setup - Casestudy

OpenNebula offers a simple but feature-rich and flexible solution to build and manage enterprise clouds and virtualized data centers. OpenNebula is designed to be simple. Simple to install, update and operate by the admins, and simple to use by end users.

Interfaces Provided by OpenNebula

OpenNebula provides many different interfaces that can be used to interact with the functionality offered to manage physical and virtual resources. There are four main different perspectives to interact with OpenNebula:

- Cloud interfaces for Cloud Consumers, like EC2 Query and EBS interfaces, and a simple Sunstone cloud user view that can be used as a self-service portal.
- Administration interfaces for Cloud Advanced Users and Operators, like a Unix-like command line interface and the powerful Sunstone GUI.
- Extensible low-level APIs for Cloud Integrators in Ruby, JAVA and XMLRPC API
- A Marketplace for Appliance Builders with a catalog of virtual appliances ready to run in OpenNebula environments.

OpenNebula Support to Cloud Consumers

OpenNebula provides a powerful, scalable and secure multi-tenant cloud platform for fast delivery and elasticity of virtual resources. Multi-tier applications can be deployed and consumed as pre-configured virtual appliances from catalogs.

- **Image Catalogs:** OpenNebula allows to store disk images in catalogs (termed datastores), that can be then used to define VMs or shared with other users. The images can be OS installations, persistent data sets or empty data blocks that are created within the datastore.
- **Network Catalogs:** Virtual networks can be also be organised in network catalogs, and provide means to interconnect virtual machines. This kind of resources can be defined as IPv4, IPv6, or mixed networks, and can be used to achieve full isolation between virtual networks.
- **VM Template Catalog:** The template catalog system allows to register virtual machine definitions in the system, to be instantiated later as virtual machine instances.
- **Virtual Resource Control and Monitoring:** Once a template is instantiated to a virtual machine, there are a number of operations that can be performed to control lifecycle of the virtual machine instances, such as migration (live and cold), stop, resume, cancel, poweroff, etc.
- **Multi-tier Cloud Application Control and Monitoring:** OpenNebula allows to define, execute and manage multi-tiered elastic applications, or services composed of interconnected Virtual Machines with deployment dependencies between them and auto-scaling rules

OpenNebula Support to Cloud Consumers

- **Users and Groups**: OpenNebula features advanced multi-tenancy with powerful users and groups management, fine-grained ACLs for resource allocation, and resource quota management to track and limit computing, storage and networking utilization.
- **Virtualization**: Various hypervisors are supported in the virtualization manager, with the ability to control the complete lifecycle of Virtual Machines and multiple hypervisors in the same cloud infrastructure.
- **Hosts**: The host manager provides complete functionality for the management of the physical hosts in the cloud.
- **Monitoring**: Virtual resources as well as hosts are periodically monitored for key performance indicators. The information can then used by a powerful and flexible scheduler for the definition of workload and resource-aware allocation policies.
- **Accounting**: A Configurable accounting system to visualize and report resource usage data, to allow their integration with chargeback and billing platforms, or to guarantee fair share of resources among users.
- **Networking**: An easily adaptable and customizable network subsystem is present in OpenNebula in order to better integrate with the specific network requirements of existing data centers and to allow full isolation between virtual machines that composes a virtualised service.
- **Storage**: The support for multiple datastores in the storage subsystem provides extreme flexibility in planning the storage backend and important performance benefits.
- **Security**: This feature is spread across several subsystems: authentication and authorization mechanisms allowing for various possible mechanisms to identify a authorize users, a powerful Access Control List mechanism allowing different role management with fine grain permission granting over any resource managed by OpenNebula, support for isolation at different levels...
- **High Availability**: Support for HA architectures and configurable behavior in the event of host or VM failure to provide easy to use and cost-effective failover solutions.
- **Clusters**: Clusters are pools of hosts that share datastores and virtual networks. Clusters are used for load balancing, high availability, and high performance computing.
- **Multiple Zones**: The Data Center Federation functionality allows for the centralized management of multiple instances of OpenNebula for scalability, isolation and multiple-site support.
- **VDCs**. An OpenNebula instance (or Zone) can be further compartmentalized in Virtual Data Centers (VDCs), which offer a fully-isolated virtual infrastructure environment where a group of

users, under the control of the group administrator, can create and manage compute, storage and networking capacity.

- **Cloud Bursting**: OpenNebula gives support to build a hybrid cloud, an extension of a private cloud to combine local resources with resources from remote cloud providers. A whole public cloud provider can be encapsulated as a local resource to be able to use extra computational capacity to satisfy peak demands.
- **App Market**: OpenNebula allows the deployment of a private centralized catalog of cloud applications to share and distribute virtual appliances across OpenNebula instances

OpenNebula Support to Cloud builders

OpenNebula offers broad support for commodity and enterprise-grade hypervisor, monitoring, storage, networking and user management services:

User Management: OpenNebula can validate users using its own internal user database based on passwords, or external mechanisms, like ssh, x509, ldap or Active Directory

Virtualization: Several hypervisor technologies are fully supported, like Xen, KVM and VMware.

Monitoring: OpenNebula provides its own customizable and highly scalable monitoring system and also can be integrated with external data center monitoring tools.

Networking: Virtual networks can be backed up by 802.1Q VLANs, ebtables, Open vSwitch or VMware networking.

Storage: Multiple backends are supported like the regular (shared or not) filesystem datastore supporting popular distributed file systems like NFS, Lustre, GlusterFS, ZFS, GPFS, MooseFS...; the VMware datastore (both regular filesystem or VMFS based) specialized for the VMware hypervisor that handle the vmdk format; the LVM datastore to store disk images in a block device form; and Ceph for distributed block device.

Databases: Aside from the original sqlite backend, mysql is also supported.

OpenNebula 4.6 Installation on Ubuntu 14.04 and Kernel Virtual Machine (KVM) Hypervisor

OpenNebula installation using Ubuntu 14.04 as Operating system and KVM as the hypervisor. The two components are present in it are Frontend and Nodes. The Frontend server execute the OpenNebula services and the Nodes functionality is to execute virtual machines. The following steps to execute both components in a single server. The machine has to support virtualization (i.e.) VT enabled system. The command to test the virtualization support is as follows

```
grep -E 'svm|vmx' /proc/cpuinfo
```

Package Layout

opennebula-common: Provides the user and common files

libopennebula-ruby: All ruby libraries

opennebula-node: Prepares a node as an opennebula-node

opennebula-sunstone: OpenNebula Sunstone Web Interface

opennebula-tools: Command Line interface

opennebula-gate: Gate server that enables communication between VMs and OpenNebula

opennebula-flow: Manages services and elasticity

opennebula: OpenNebula Daemon

Step 1. Installation in the Frontend

1. Add the repository

```
sudo wget -q -O- http://downloads.opennebula.org/repo/Ubuntu/repo.key | apt-key add -
sudo echo "deb http://downloads.opennebula.org/repo/Ubuntu/14.04 stable opennebula" > /etc/apt/sources.list.d/opennebula.list
```

2. Install the required packages

```
sudo apt-get update
sudo apt-get install opennebula opennebula-sunstone nfs-kernel-server
```

3. Configure and start the services

There are two main processes that must be started, the main OpenNebula daemon: **oned**, and the graphical user interface: **sunstone**.

Sunstone listens only in the loopback interface by default for security reasons. To change it edit /etc/one/sunstone-server.conf and

change :host: 127.0.0.1 to :host: 0.0.0.0.

The command to restart the Sunstone:

```
sudo /etc/init.d/opennebula-sunstone restart
```

4. Configure Network File Service (NFS) (This is not needed if both Frontend and Nodes are in the same machine)

Export /var/lib/one/ from the frontend to the worker nodes. To do so add the following to the /etc/exports file in the frontend:

```
/var/lib/one/ *(rw,sync,no_subtree_check,root_squash)
```

Refresh the NFS export by the following command

```
sudo service nfs-kernel-server restart
```

5. Configure SSH public key

```
sudo su – oneadmin
cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
```

Add the following snippet to ~/.ssh/config so it doesn't prompt to add the keys to the known_hosts file:.

```
$ cat << EOT > ~/.ssh/config
Host *
   StrictHostKeyChecking no
   UserKnownHostsFile /dev/null
EOT
$ chmod 600 ~/.ssh/config
```

Step 2. Installation in the Nodes

1. Install the required packages

```
sudo apt-get install opennebula-node nfs-common bridge-utils
```

2. Configure the network

In DHCP,

edit /etc/network/interfaces

```
auto lo
iface lo inet loopback
auto br0
```

```
iface br0 inet dhcp
    bridge_ports eth0
    bridge_fd 9
    bridge_hello 2
    bridge_maxage 12
    bridge_stp off
```

In case of static IP address

```
auto lo
iface lo inet loopback

auto br0
iface br0 inet static
    address 192.168.0.10
    network 192.168.0.0
    netmask 255.255.255.0
    broadcast 192.168.0.255
    gateway 192.168.0.1
    bridge_ports eth0
    bridge_fd 9
    bridge_hello 2
    bridge_maxage 12
    bridge_stp off
```

Restart the network

```
sudo /etc/init.d/networking restart
```

3. Configure NFS (This is not needed if both Frontend and Nodes are in the same machine)

edit the file /etc/fstab as

```
<Frontend IP>:/var/lib/one/  /var/lib/one/  nfs   soft,intr,rsize=8192,wsize=8192,noauto
```

Mount the NFS

```
sudo  mount /var/lib/one
```

/* if there is any problem , then it will be firewall issue

4. Configure Qemu

oneadmin user must be able to manage libvirt as root

```
# cat << EOT > /etc/libvirt/qemu.conf
user = "oneadmin"
group = "oneadmin"
dynamic_ownership = 0
EOT
```

Restart libvirt

```
sudo service libvirt-bin restart
```

Step 3: Start the sunstone from the web browser

```
http://frontend:9869
```

oneadmin password is available in the file ~/.one/one_auth

1. From the command line, to login as oneadmin in the Frontend , execute

```
sudo su - oneadmin
```

2. Adding Host

To start running VMs, you should first register a worker node for OpenNebula.

```
onehost create localhost -i kvm -v kvm -n dummy
```

Run **onehost list** - Try these command until it's on. If there is any failure, look at /var/log/one/oned.log

3. Adding Virtual resources

To create networks, create a network template file "mynetwork.one" and edit the file as follows:

```
NAME = "private"
TYPE = FIXED
BRIDGE = br0
LEASES = [ IP=192.168.0.100 ]
LEASES = [ IP=192.168.0.101 ]
LEASES = [ IP=192.168.0.102 ]
```

Create resources in opennebula

```
$ onevnet create mynetwork.one
$ oneimage create --name "CentOS-6.5_x86_64" \
   --path "http://appliances.c12g.com/CentOS-6.5/centos6.5.qcow2.gz" \
   --driver qcow2 \
   --datastore default
$ onetemplate create --name "CentOS-6.5" --cpu 1 --vcpu 1 --memory 512 \
   --arch x86_64 --disk "CentOS-6.5_x86_64" --nic "private" --vnc \
   --ssh
```

Monitor the resources are created by running the command **oneimage list**

In order to dynamically add ssh keys to Virtual Machines we must add our ssh key to the user template, by editing the user template:

```
$  EDITOR = vi oneuser update oneadmin
```

Add a new line to the template (cat ~/.ssh/id_dsa.pub)

```
SSH_PUBLIC_KEY="ssh-dss AAAAB3NzaC1kc3MAAACBANBWTQmm4Gt..."
```

4. To run a virtual machine

```
$ onetemplate instantiate "Centos-6.5"  --name  "My Scratch VM"
```

To test, execute onevm list , VM going from Pending to Prolog to Running. If it fails, check the log as /var/log/one/<VM_ID>/vm.log

OPEN NEBULA – FRONT END INSTALLATION

AIM:

To install the Open Nebula in Ubuntu 14.04 Operating System.

PROCEDURE:

NETWORK CONFIGURATION

Step 1: Note the IP address of front end machine and cluster node machine.

```
$ifconfig
```

Step 2: Verify the connectivity with the cluster node using ping.

```
$ping <ipaddress of cluster node>
```

Step 3: Set the Host Name as 'frontend' in /etc/hostname file

```
$sudo nano /etc/hostname
```

Step 4: Set the Known Hosts in /etc/hosts file

```
$sudo nano /etc/hosts
```

127.0.0.1	**localhost**
192.168.116.00	**frontend**
192.168.116.00	**clusterend**

RESTART THE SYSTEM

CLOUD USER ACCOUNT CREATION

Step 5: Create a user with name, group and password

```
sudo mkdir –p /srv/cloud
sudo groupadd cloud
sudo useradd –u 1001 –g cloud –m cloudadmin –d /srv/cloud/one – s/bin/bash
sudo passwd cloudadmin
sudo chown –R cloudadmin:cloud /srv/cloud
sudo adduser cloudadmin sudo
```

Logout and login to "cloudadmin" user

INSTALLATION OF NFS SERVER

Step 6: Install the NFS server

```
sudo apt-get install nfs-kernel-server
sudo nano /etc/exports
```

Append cluster node ip-address to /etc/exports file

```
/srv/cloud/one <ip address>/24(rw,fsid=0,nohide,sync,root_squash,no_ subtree_check)
```

Restart NFS Server

```
sudo /etc/init.d/nfs-kernel-server restart
```

Successful installation of NFS server will display the following

```
Stopping NFS Kernel Daemon [OK]
Unexporting directories for NFS Kernel Daemon [OK]
Exporting directories for NFS Kernel Daemon[OK]
Starting NFS Kernel Daemon [OK]
```

CONFIGURATION OF SECURE SHELL (SSH)

Step 7: Secure Shell interface is configured for passwordless communication between front end & cluster node

```
sudo apt-get install openssh-server
```

Edit /etc/ssh/sshd_config with the following command

```
$ sudo cp /etc/ssh/sshd_config ~
$ sudo nano /etc/ssh/sshd_config
```

Modify the following lines in sshd-config file

```
PermitRootLogin no
StrictModes no
```

Restart the SSH daemon

```
$sudo restart ssh
```

Successful restart of SSH daemon will display the following lines

```
ssh start/running process 3347
```

Confirm the current working directory is /srv/cloud/one

```
$pwd
$mkdir .ssh
$cd .ssh
.ssh$ ssh-keygen -t rsa
.ssh$ ls
```

Step 8: Send the public key to the cluster node.

```
.ssh $ sudo scp id_rsa.pub cloudadmin@clusterend: /srv/cloud/one/.ssh/id_rsa_localbox.pub
```

Step 9: Wait for cluster node to complete Step-8

Change to /srv/cloud/one directory and check SSH using following command

```
.ssh $ cd ..
$ ssh cloudadmin@clusterend
$exit
```

INSTALLATION OF DEPENDENCY SOFTWARE

Step 10: Install dependency softwares for OpenNebula installation

```
sudo apt-get install libsqlite3-dev
sudo apt-get install libxmlrpc-core-c3
sudo apt-get install g++
sudo apt-get install ruby1.9.3
sudo apt-get install openssl-ruby1.9
sudo apt-get install libssl-dev
sudo apt-get install ruby-dev
sudo apt-get install libxml2-dev
sudo apt-get install libmysqlclient-dev
sudo apt-get install libmysql++-dev
sudo apt-get install sqlite3
sudo apt-get install libexpat1-dev
sudo apt-get install rake
sudo apt-get install gems
sudo apt-get install genisoimage
sudo apt-get install scons
```

```
sudo apt-get install xmlrpc-c++
sudo apt-get install libxmlrpc-c++
```

INSTALLING DEPENDENCY GEMS

```
sudo gem install libxml-ruby
sudo gem install nokogiri
sudo gem install rake
sudo gem install sinatra
sudo gem install json
```

INSTALLATION OF OPENNEBULA

Step 10: Verify the present working directory as /srv/cloud/one and install open-nebula

```
$ pwd
$ sudo tar xvzf Downloads/opennebula-4.2.0.tar.gz
$sudo chown cloudadmin:<groupname> -R opennebula-4.2.0
$sudo chmod 755 –R opennebula-4.2.0
$sudo umount .gvfs
$ cd opennebula-4.2.0
~/opennebula-4.2.0$ scons sqlite=yes mysql=no
```

Successful initialization using scons will display the following message

```
- scons done building targets.
```

```
/opennebula-4.2.0$ sudo ./install.sh -u cloudadmin -g <groupname> -d /srv/cloud/one
/opennebula-4.2.0$ cd /usr/share
/usr/share$ mkdir one
```

Set the environment variables using openenv.sh at /srv/cloud/one directory

```
/usr/share$ cd /srv/cloud/one
$ pwd
$ sudo cp /home/student/Downloads/openenv.sh .
$ sudo chmod 755 –R openenv.sh
$ source openenv.sh
```

```
$ echo $ONE_LOCATION
```

Create .one file in the /srv/cloud/one directory for authentication

```
$ sudo mkdir .one
$ cd .one
/.one$ ls
/.one$ sudo nano one_auth
```

Type the user name and password in /srv/cloud/one/.one for authentication.

```
cloudadmin:cloudadmin
```

Configure etc/oned.conf .

```
/.one$ cd ..
$ sudo nano etc/oned.conf
```

Check whether the following lines is uncommented.

```
DB = [ backend = "sqlite" ]
```

Specify the emulator in vmm_exec_kvm.conf file

```
$sudo nano /srv/cloud/one/etc/vmm_exec/vmm_exec_kvm.conf
```

Comment the first line and add the next line.

```
#EMULATOR = /usr/libexec/qemu-kvm
EMULATOR = /usr/bin/kvm
```

Verify the current working directory is /srv/cloud/one. Start scheduler and sunstone-server.

```
$ pwd
$ one stop
$ one start
$ sunstone-server start
```

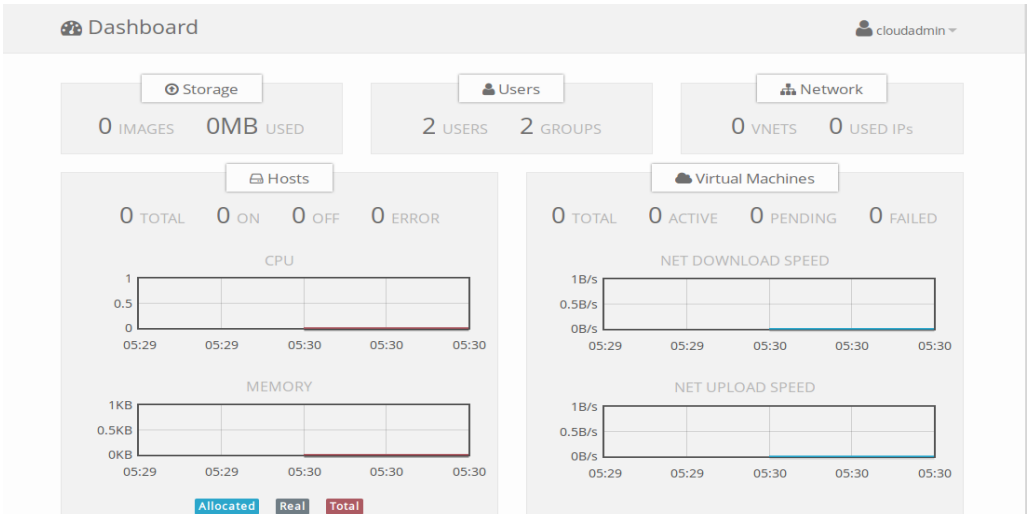
Successful installation of sun-stone server will display the following message.

```
VNC proxy started
```

sunstone-server started

To open the sunstone server – open a web browser and type http://localhost:9869/

OUTPUT SCREENSHOTS:



RESULT:

Thus, the Open Nebula in Ubuntu 14.04 Operating System is executed successfully.

Open stack installation in a Single machine

AIM: To install Open Stack in Ubuntu 14.04 Operating System.

Prerequisites:

OS **:** Ubuntu 14.04 LTS 64 bit

RAM **:** 8 GB

Processor **:** i3

Install the Ubuntu 14.04 LTS server. Once the server is installed and from the command prompt, enables the desktop using the following command

```
$ sudo apt-get install   ubuntu-gnome-desktop
```

1. Update the installed packages

```
$ sudo      apt-get      update
```

1. Install git package

```
$ sudo apt-get install git
```

2. Install the devstack packages

```
$ git clone https://git.openstack.org/openstack-dev/devstack.git -b stable/kilo
```

4. Chage the directory to devstack

```
$ cd devstac
```

5. Script to install openstack is in **stackrc** file and edit the file as follows

```
$ sudo nano stackrc
GIT_BASE=${GIT_BASE:-https://www.github.com}
```

6. stack the file devstack$

```
./stack.sh
```

Four times ask to enter password during stack operation. If there is any error occurs during stack, unstack the file devstack$ ./unstack.sh

7. If there is any occur in the mysql installation, run the following command

```
$ sudo dpkg –reconfigure mysql-server-5.5
```

Edit the local.conf file is as follows

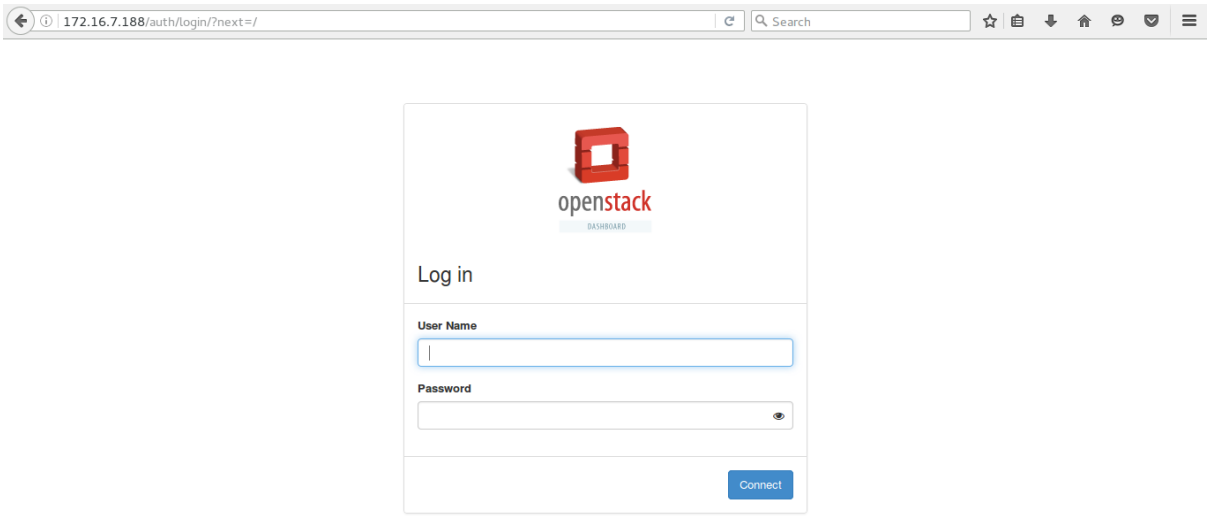
```
$ nano local.conf
enable-service s-proxy s-object s-container s-account FIXED-RANGE=<network ip
address>/24 FLOATING_RANGE=10.0.0.0/24
HOST_IP=<your machine ip address>
NETWORK_GATEWAY=<your gateway address>
Q_FLOATING_ALLOCATION_POOL=START=<start ip address>,END=<end ip address>
```

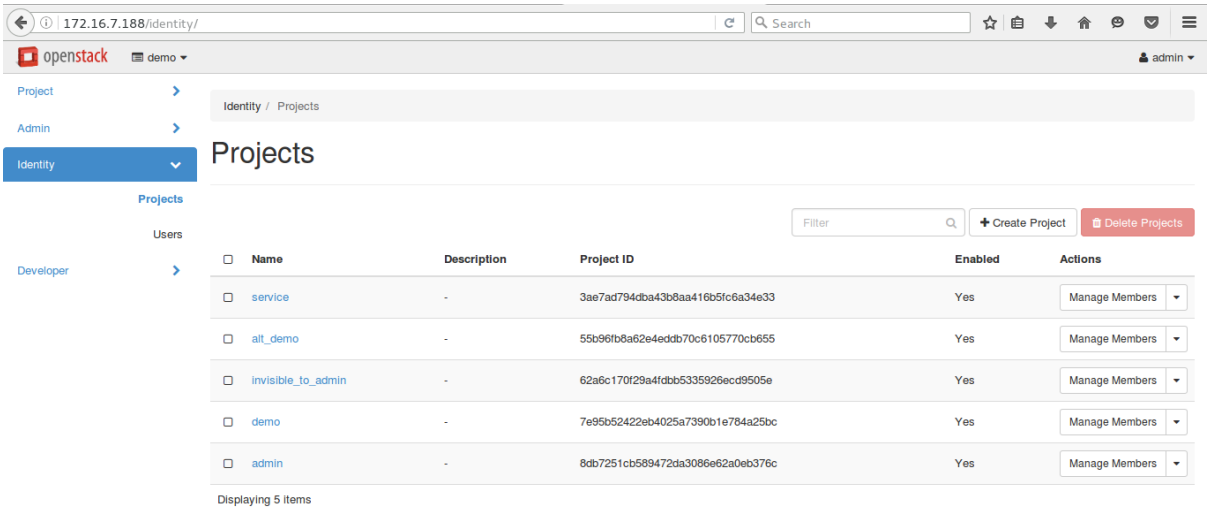
Stack the file (i.e) ./stack.sh

Openstack dashboard view it in browser

```
http://<machine ip address got from stack >
```

SCREENSHOTS:





RESULT: Thus, the open stack is implemented in Ubuntu 14.04 Operating System.

Hadoop - Case Study

Big Data - Big Data is a collection of large datasets that cannot be processed using traditional computing techniques. It is not a single technique or a tool, rather it involves many areas of business and technology.

Types of data under Big Data

Big data involves the data produced by different devices and applications.

Black Box Data: It is a component of helicopter, airplanes, and jets, etc. It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.

Social Media Data: Social media such as Facebook and Twitter hold information and the views posted by millions of people across the globe.

Stock Exchange Data: The stock exchange data holds information about the 'buy' and 'sell' decisions made on a share of different companies made by the customers.

Power Grid Data: The power grid data holds information consumed by a particular node with respect to a base station.

Transport Data: Transport data includes model, capacity, distance and availability of a vehicle.

Search Engine Data: Search engines retrieve lots of data from different databases.

Thus Big Data includes huge volume, high velocity, and extensible variety of data. The data in it will be of three types.

- **Structured data:** Relational data.
- **Semi Structured data:** XML data.
- **Unstructured data:** Word, PDF, Text, Media Logs.

Benefits of Big Data

- Using the information kept in the social network like Facebook, the marketing agencies are learning about the response for their campaigns, promotions, and other advertising mediums.
- Using the information in the social media like preferences and product perception of their consumers, product companies and retail organizations are planning their production.

- Using the data regarding the previous medical history of patients, hospitals are providing better and quick service.

Big Data Technologies

Big data technologies are important in providing more accurate analysis, which may lead to more concrete decision-making resulting in greater operational efficiencies, cost reductions, and reduced risks for the business. To harness the power of big data, you would require an infrastructure that can manage and process huge volumes of structured and unstructured data in real time and can protect data privacy and security.

There are various technologies in the market from different vendors including Amazon, IBM, Microsoft, etc., to handle big data. While looking into the technologies that handle big data, we examine the following two classes of technology:

Operational Big Data

These include systems like MongoDB that provide operational capabilities for real-time, interactive workloads where data is primarily captured and stored.

NoSQL Big Data systems are designed to take advantage of new cloud computing architectures that have emerged over the past decade to allow massive computations to be run inexpensively and efficiently. This makes operational big data workloads much easier to manage, cheaper, and faster to implement.

Some NoSQL systems can provide insights into patterns and trends based on real-time data with minimal coding and without the need for data scientists and additional infrastructure.

Analytical Big Data

These includes systems like Massively Parallel Processing (MPP) database systems and MapReduce that provide analytical capabilities for retrospective and complex analysis that may touch most or all of the data.

MapReduce provides a new method of analyzing data that is complementary to the capabilities provided by SQL, and a system based on MapReduce that can be scaled up from single servers to thousands of high and low end machines.

Big Data Challenges

The major challenges associated with big data are as follows such as Capturing data, Curation, Storage, Searching , Sharing , Transfer, Analysis and Presentation.

Hadoop 2.6 Installation on Ubuntu 14.04 (Single-Node Cluster)

1. Update the source list

```
$ sudo apt-get update
```

2. Install Java

```
$ sudo apt-get install openjdk-7-jdk
$ java -version
```

3. Adding a dedicated Hadoop user

```
$ sudo addgroup hadoop
$ sudo adduser --ingroup hadoop hduser
```

4. Installing ssh

```
$ sudo apt-get install ssh
```

5. Create and Setup SSH certificates

```
$ su hduser
$ ssh-keygen -t rsa -P " "
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

To check the ssh works ,

```
$ ssh localhost
```

6. Install Hadoop

```
$wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz
$ tar xvzf hadoop-2.6.0.tar.gz
$ sudo mv * /usr/local/hadoop
```

hduser make root privilege

```
$ sudo su hduser
$ sudo mv * /usr/local/hadoop
$ sudo chown -R hduser:hadoop /usr/local/hadoop
```

7. Setup configuration files

Check the java installed location

Append to the end of ~/.bashrc

$ vi ~/.bashrc

```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END
```

$ source ~/.bashrc

To check JAVA_HOME

$ javac -version

$ which javac

$ readlink -f /usr/bin/javac

2. Edit the hadoop environmental variable

$ vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

3. Edit the core-site.xml

$ sudo mkdir -p /app/hadoop/tmp

$ sudo chown hduser:hadoop /app/hadoop/tmp

$ vi /usr/local/hadoop/etc/hadoop/core-site.xml

```
<configuration>
 <property>
```

```
 <name>hadoop.tmp.dir</name>
 <value>/app/hadoop/tmp</value>
 <description>A base for other temporary directories.</description>
</property>
<property>
 <name>fs.default.name</name>
 <value>hdfs://localhost:54310</value>
 <description>The name of the default file system.  A URI whose
 scheme and authority determine the FileSystem implementation.  The
 uri's scheme determines the config property (fs.SCHEME.impl) naming
 the FileSystem implementation class.  The uri's authority is used to
 determine the host, port, etc. for a filesystem.</description>
</property>
</configuration>
```

4. Edit mapred-site.xml

```
cp  /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
        /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

Edit the mapred-site.xml between <configuration> </configuration>

```
<configuration>
 <property>
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
  <description>The host and port that the MapReduce job tracker runs
  at.  If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
 </property>
</configuration>
```

5. Configure /usr/local/hadoop/etc/hadoop/hdfs-site.xml

```
hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
```

```
hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
hduser@laptop:~$ sudo chown -R hduser:hadoop /usr/local/hadoop_store
```

```
$ vi /usr/local/hadoop/etc/hadoop/hdfs-site.xml
<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
</configuration>
```

Format the new Hadoop filesystem
/usr/local/hadoop_store/hdfs/namenode

```
$ hadoop  namenode  -format
```

Start Hadoop

```
/usr/local/hadoop/sbin$ cd /usr/local/hadoop/sbin
/usr/local/hadoop/sbin S ls
/usr/local/hadoop/sbin $ sudo  su  hduser
/usr/local/hadoop/sbin $ start-all.sh
```

To check

```
/usr/local/hadoop/sbin S jps
```

(or)

```
$ netstat -plten | grep java
```

To stop the hadoop

```
/usr/local/hadoop/sbin $ stop-all.sh
```

Web interface

http://localhost:50070/

Secondary Namenode

http://localhost:50090/status.jsp

Hadoop Distributed File System (HDFS) shell commands

1. To list the contents of a directory

```
$ hadoop fs -ls /user
Found 8 items
drwxr-xr-x   - cloudera cloudera          0 2015-03-28 23:43 /user/cloudera
drwxr-xr-x   - hdfs     supergroup        0 2015-03-14 20:11 /user/hdfs
drwxr-xr-x   - mapred   hadoop            0 2015-03-15 14:08 /user/history
drwxrwxrwx   - hive     hive              0 2014-12-18 04:33 /user/hive
drwxrwxr-x   - hue      hue               0 2015-03-21 15:34 /user/hue
drwxrwxrwx   - oozie    oozie             0 2014-12-18 04:34 /user/oozie
drwxr-xr-x   - sample   sample            0 2015-03-14 22:05 /user/sample
drwxr-xr-x   - spark    spark             0 2014-12-18 04:34 /user/spark
```

2. To view the contents of a file

```
$ hadoop fs -cat /user/hdfs/dir1/myfile
This is myfile
```

3. To create a directory in HDFS

```
$ sudo su hdfs
bash-4.1$ hadoop fs -mkdir /user/hdfs/dir1
```

4. To copy src files from local file system to the Hadoop data file system

```
$ hadoop fs -put <local_src> <HDFS_dest_Path>
$ whoami
hdfs
$ hadoop fs -put myfile /user/hdfs/dir1
```

5. To copy files from the Hadoop data file system to the local file system

```
$ hadoop fs -get /user/hdfs/dir1/myfile .
```

6. To copy a file from source to destination

```
$ hadoop fs -cp /user/hdfs/dir1/myfile /user/hdfs/dir2
$ hadoop fs -ls /user/hdfs/dir2/myfile
-rw-r--r-- 1 hdfs supergroup 7 2015-03-29 11:43 /user/hdfs/dir2/myfile
```

7. To copy a file from Local file system to HDFS

```
hadoop fs -copyFromLocal <localsrc> URI
$ pwd
/home/cloudera/workspace/temp
$ ls
myfile myfile2
$ hadoop fs -copyFromLocal myfile2 /user/hdfs/dir2/
$ hadoop fs -cat /user/hdfs/dir2/myfile2
This is myfile2
```

8. To copy a file to Local file system from HDFS

```
$ hadoop fs -copyToLocal /user/hdfs/dir1/myfile myfile3
```

9. To remove a file from HDFS

```
$ hadoop fs  -ls  /user/hdfs/dir2
Found 2 items
-rw-r--r--   1 hdfs supergroup          7 2015-03-29 11:43 /user/hdfs/dir2/myfile
-rw-r--r--   1 hdfs supergroup         16 2015-03-29 11:52 /user/hdfs/dir2/myfile2
$ hadoop  fs  -rm   /user/hdfs/dir2/myfile2
$ hadoop  fs  -ls   /user/hdfs/dir2
-rw-r--r--   1 hdfs supergroup          7 2015-03-29 11:43 /user/hdfs/dir2/myfile
```

10. To remove a directory from HDFS

```
bash-4.1$ hadoop fs -ls /user/hdfs/
Found 3 items
drwxr-xr-x   - hdfs supergroup          0 2015-03-28 14:08 /user/hdfs/.Trash
drwxr-xr-x   - hdfs supergroup          0 2015-03-29 10:55 /user/hdfs/dir1
drwxr-xr-x   - hdfs supergroup          0 2015-03-29 12:12 /user/hdfs/dir2
bash-4.1$ hadoop fs -rm -r /user/hdfs/dir2
bash-4.1$ hadoop fs -ls /user/hdfs/
drwxr-xr-x   - hdfs supergroup          0 2015-03-28 14:08 /user/hdfs/.Trash
drwxr-xr-x   - hdfs supergroup          0 2015-03-29 10:55 /user/hdfs/dir1
```

MapReduce Application - Word Count

```
package org.myorg;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
public class WordCount {
  public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text,
IntWritable>
```

```
{
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();
   public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter
reporter) throws IOException {
     String line = value.toString();
     StringTokenizer tokenizer = new StringTokenizer(line);
     while (tokenizer.hasMoreTokens()) {
       word.set(tokenizer.nextToken());
       output.collect(word, one);
     }
   }
 }
 public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text,
IntWritable> {
   public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output,
Reporter reporter) throws IOException {
     int sum = 0;
     while (values.hasNext()) {
       sum += values.next().get();
     }
     output.collect(key, new IntWritable(sum));
   }
 }
 public static void main(String[] args) throws Exception {
   JobConf conf = new JobConf(WordCount.class);
   conf.setJobName("wordcount");
   conf.setOutputKeyClass(Text.class);
   conf.setOutputValueClass(IntWritable.class);
   conf.setMapperClass(Map.class);
   conf.setCombinerClass(Reduce.class);
   conf.setReducerClass(Reduce.class);
   conf.setInputFormat(TextInputFormat.class);
```

```
    conf.setOutputFormat(TextOutputFormat.class);
    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));
    JobClient.runJob(conf);
  }
}
```

HADOOP MAPREDUCE PROGRAM

AIM:

To execute a Hadoop Map Reduce Program – ProcessUnits.java in Ubuntu 14.04.

PROCEDURE:

1. Create the following directory :

```
$ sudo mkdir /home/hadoop
```

2. Create ProcessUnits.java and write the **program.**

3. Create the following directory

```
$ sudo mkdir /home/hadoop/units
```

4. Compile java program using the following commands

```
$ sudo javac -classpath /usr/local/hadoop/share/hadoop/common/hadoop-common-
2.6.0.jar:/usr/local/hadoop/share/hadoop/share/hadoop/common/lib/hadoop-annotations-
2.6.0.jar:/usr/local/hadoop/share/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-core-
2.6.0.jar -d units ProcessUnits.java
```

5. Create jar file as follows

```
$ jar -cvf units.jar -C units/ .
```

6.The following command is used to create an input directory in HDFS.

```
$ /usr/local/hadoop/bin/hadoop fs -mkdir inputdir
```

6. Create sample.txt in /home/hadoop

```
1979   23   23   2   43   24   25   26   26   26   26   25   26  25
1980   26   27   28  28   28   30   31   31   31   30   30   30  29
1981   31   32   32  32   33   34   35   36   36   34   34   34  34
1984   39   38   39  39   39   41   42   43   40   39   38   38  40
1985   38   39   39  39   39   41   41   41   00   40   39   39  45
```

7. The following command is used to copy the input file named **sample.txt** in the input directory of HDFS.

```
$ /usr/local/hadoop/bin/hadoop fs -put /home/hadoop/sample.txt  /home/hadoop/inputdir
```

```
export HADOOP_OPTS="$HADOOP_OPTS -Djava.library.path=/usr/local/hadoop/lib/native"
```

8. The following command is used to verify the files in the input directory.

```
$ /usr/local/hadoop/bin/hadoop fs -ls /home/hadoop/inputdir
```

If the datanode is not running, do the following command one by one

```
sudo rm -r /usr/local/hadoop_store/hdfs/*
```

sudo chmod -R 755 /usr/local/hadoop_store/hdfs

/usr/local/hadoop/bin/hadoop namenode -format

start-all.sh

jps

9. The following command is used to run the Eleunit_max application by taking the input files from the input directory.

/usr/local/hadoop/bin/hadoop jar units.jar hadoop.ProcessUnits /home/hadoop/inputdir outputdir

10. To list the output directory

/usr/local/hadoop/bin/hadoop fs -ls /home/hadoop/outputdir

11. The following command is used to see the output in **Part-00000** file. This file is generated by HDFS.

/usr/local/hadoop/bin/hadoop fs -cat outputdir/part-00000

The output generated by the MapReduce program is as follows

1981 34
1984 40
1985 45

The following command is used to copy the output folder from HDFS to the local file system for analyzing.

/usr/local/hadoop/bin/hadoop fs -cat outputdir/part-00000/bin/hadoop dfs get output_dir /home/hadoop

PROGRAM:

```
package hadoop;

import java.util.*;

import java.io.IOException;

import java.io.IOException;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapred.*;
```

```
import org.apache.hadoop.util.*;
public class ProcessUnits
{
  //Mapper class
  public static class E_EMapper extends MapReduceBase implements
  Mapper<LongWritable ,/*Input key Type */
  Text,              /*Input value Type*/
  Text,              /*Output key Type*/
  IntWritable>       /*Output value Type*/
  {
    //Map function
    public void map(LongWritable key, Text value,
    OutputCollector<Text, IntWritable> output,
    Reporter reporter) throws IOException
    {
      String line = value.toString();
      String lasttoken = null;
      StringTokenizer s = new StringTokenizer(line,"\t");
      String year = s.nextToken();
   while(s.hasMoreTokens())
        {
          lasttoken=s.nextToken();
        }
      int avgprice = Integer.parseInt(lasttoken);
      output.collect(new Text(year), new IntWritable(avgprice));
    }
  }
 //Reducer class
```

```
  public static class E_EReduce extends MapReduceBase implements
  Reducer< Text, IntWritable, Text, IntWritable >
  {
          //Reduce function
    public void reduce( Text key, Iterator <IntWritable> values,
      OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException
      {
        int maxavg=30;
        int val=Integer.MIN_VALUE;

        while (values.hasNext())
        {
          if((val=values.next().get())>maxavg)
          {
            output.collect(key, new IntWritable(val));
          }
        }

      }
  }
//Main function
  public static void main(String args[])throws Exception
  {
    JobConf conf = new JobConf(ProcessUnits.class);
   conf.setJobName("max_eletricityunits");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(E_EMapper.class);
```

```
    conf.setCombinerClass(E_EReduce.class);
    conf.setReducerClass(E_EReduce.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    FileInputFormat.setInputPaths(conf, new Path(args[0]));
   FileOutputFormat.setOutputPath(conf, new Path(args[1]));
    JobClient.runJob(conf);
  }
}
```

OUTPUT:

/usr/local/hadoop/bin/hadoop jar units.jar hadoop.ProcessUnits /home/hadoop/inputdir outputdir

16/09/12 15:07:04 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

16/09/12 15:07:04 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id

16/09/12 15:07:04 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=

16/09/12 15:07:04 INFO jvm.JvmMetrics: Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already initialized

16/09/12 15:07:04 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.

16/09/12 15:07:05 INFO mapred.FileInputFormat: Total input paths to process : 1

16/09/12 15:07:05 INFO mapreduce.JobSubmitter: number of splits:1

16/09/12 15:07:05 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local925768423_0001

16/09/12 15:07:06 INFO mapreduce.Job: The url to track the job: http://localhost:8080/

16/09/12 15:07:06 INFO mapreduce.Job: Running job: job_local925768423_0001

16/09/12 15:07:06 INFO mapred.LocalJobRunner: OutputCommitter set in config null

16/09/12 15:07:06 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter

16/09/12 15:07:06 INFO mapred.LocalJobRunner: Waiting for map tasks

16/09/12 15:07:06 INFO mapred.LocalJobRunner: Starting task: attempt_local925768423_0001_m_000000_0

16/09/12 15:07:06 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]

16/09/12 15:07:06 INFO mapred.MapTask: Processing split: hdfs://localhost:54310/home/hadoop/inputdir/sample.txt:0+210

16/09/12 15:07:06 INFO mapred.MapTask: numReduceTasks: 1

16/09/12 15:07:06 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)

16/09/12 15:07:06 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100

16/09/12 15:07:06 INFO mapred.MapTask: soft limit at 83886080

16/09/12 15:07:06 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600

16/09/12 15:07:06 INFO mapred.MapTask: kvstart = 26214396; length = 6553600

16/09/12 15:07:06 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer

16/09/12 15:07:06 INFO mapred.LocalJobRunner:

16/09/12 15:07:06 INFO mapred.MapTask: Starting flush of map output

16/09/12 15:07:06 INFO mapred.MapTask: Spilling map output

16/09/12 15:07:06 INFO mapred.MapTask: bufstart = 0; bufend = 45; bufvoid = 104857600

16/09/12 15:07:06 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 26214380(104857520); length = 17/6553600

16/09/12 15:07:06 INFO mapred.MapTask: Finished spill 0

16/09/12 15:07:06 INFO mapred.Task: Task:attempt_local925768423_0001_m_000000_0 is done. And is in the process of committing

16/09/12 15:07:06 INFO mapred.LocalJobRunner: hdfs://localhost:54310/home/hadoop/inputdir/sample.txt:0+210

16/09/12 15:07:06 INFO mapred.Task: Task 'attempt_local925768423_0001_m_000000_0' done.

16/09/12 15:07:06 INFO mapred.LocalJobRunner: Finishing task: attempt_local925768423_0001_m_000000_0

16/09/12 15:07:06 INFO mapred.LocalJobRunner: map task executor complete.

16/09/12 15:07:06 INFO mapred.LocalJobRunner: Waiting for reduce tasks

16/09/12 15:07:06 INFO mapred.LocalJobRunner: Starting task: attempt_local925768423_0001_r_000000_0

16/09/12 15:07:06 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]

16/09/12 15:07:06 INFO mapred.ReduceTask: Using ShuffleConsumerPlugin: org.apache.hadoop.mapreduce.task.reduce.Shuffle@70b02e04

16/09/12 15:07:06 INFO reduce.MergeManagerImpl: MergerManager: memoryLimit=363285696, maxSingleShuffleLimit=90821424, mergeThreshold=239768576, ioSortFactor=10, memToMemMergeOutputsThreshold=10

16/09/12 15:07:06 INFO reduce.EventFetcher: attempt_local925768423_0001_r_000000_0 Thread started: EventFetcher for fetching Map Completion Events

16/09/12 15:07:06 INFO reduce.LocalFetcher: localfetcher#1 about to shuffle output of map attempt_local925768423_0001_m_000000_0 decomp: 35 len: 39 to MEMORY

16/09/12 15:07:06 INFO reduce.InMemoryMapOutput: Read 35 bytes from map-output for attempt_local925768423_0001_m_000000_0

16/09/12 15:07:06 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-output of size: 35, inMemoryMapOutputs.size() -> 1, commitMemory -> 0, usedMemory ->35

16/09/12 15:07:06 INFO reduce.EventFetcher: EventFetcher is interrupted.. Returning

16/09/12 15:07:06 INFO mapred.LocalJobRunner: 1 / 1 copied.

16/09/12 15:07:06 INFO reduce.MergeManagerImpl: finalMerge called with 1 in-memory map-outputs and 0 on-disk map-outputs

16/09/12 15:07:06 INFO mapred.Merger: Merging 1 sorted segments

16/09/12 15:07:06 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 28 bytes

16/09/12 15:07:06 INFO reduce.MergeManagerImpl: Merged 1 segments, 35 bytes to disk to satisfy reduce memory limit

16/09/12 15:07:06 INFO reduce.MergeManagerImpl: Merging 1 files, 39 bytes from disk

16/09/12 15:07:06 INFO reduce.MergeManagerImpl: Merging 0 segments, 0 bytes from memory into reduce

16/09/12 15:07:06 INFO mapred.Merger: Merging 1 sorted segments

16/09/12 15:07:06 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 28 bytes

16/09/12 15:07:06 INFO mapred.LocalJobRunner: 1 / 1 copied.

16/09/12 15:07:07 INFO mapreduce.Job: Job job_local925768423_0001 running in uber mode : false

16/09/12 15:07:07 INFO mapreduce.Job: map 100% reduce 0%

16/09/12 15:07:07 INFO mapred.Task: Task:attempt_local925768423_0001_r_000000_0 is done. And is in the process of committing

16/09/12 15:07:07 INFO mapred.LocalJobRunner: 1 / 1 copied.

16/09/12 15:07:07 INFO mapred.Task: Task attempt_local925768423_0001_r_000000_0 is allowed to commit now

16/09/12 15:07:07 INFO output.FileOutputCommitter: Saved output of task 'attempt_local925768423_0001_r_000000_0' to hdfs://localhost:54310/user/hduser/outputdir/_temporary/0/task_local925768423_0001_r_000000

16/09/12 15:07:07 INFO mapred.LocalJobRunner: reduce > reduce

16/09/12 15:07:07 INFO mapred.Task: Task 'attempt_local925768423_0001_r_000000_0' done.

16/09/12 15:07:07 INFO mapred.LocalJobRunner: Finishing task: attempt_local925768423_0001_r_000000_0

16/09/12 15:07:07 INFO mapred.LocalJobRunner: reduce task executor complete.

16/09/12 15:07:08 INFO mapreduce.Job: map 100% reduce 100%

16/09/12 15:07:08 INFO mapreduce.Job: Job job_local925768423_0001 completed successfully

16/09/12 15:07:08 INFO mapreduce.Job: Counters: 38

File System Counters

FILE: Number of bytes read=6722

FILE: Number of bytes written=506757

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=420

HDFS: Number of bytes written=24

HDFS: Number of read operations=15

HDFS: Number of large read operations=0

HDFS: Number of write operations=4

Map-Reduce Framework

Map input records=5

Map output records=5

Map output bytes=45

Map output materialized bytes=39

Input split bytes=106

Combine input records=5

Combine output records=3

Reduce input groups=3

Reduce shuffle bytes=39

Reduce input records=3

Reduce output records=3

Spilled Records=6

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=55

CPU time spent (ms)=0

Physical memory (bytes) snapshot=0

Virtual memory (bytes) snapshot=0

Total committed heap usage (bytes)=326377472

Shuffle Errors

BAD_ID=0

CONNECTION=0

IO_ERROR=0

WRONG_LENGTH=0

WRONG_MAP=0

WRONG_REDUCE=0

File Input Format Counters

Bytes Read=210

File Output Format Counters

Bytes Written=24

hduser@frontend:/home/hadoop$ /usr/local/hadoop/bin/hadoop fs -ls /home/hadoop

16/09/12 15:07:37 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Found 3 items

drwxr-xr-x - hduser supergroup 0 2016-08-31 10:47 /home/hadoop/inputdir

drwxr-xr-x - hduser supergroup 0 2016-08-31 10:55 /home/hadoop/output_dir

drwxr-xr-x - hduser supergroup 0 2016-08-31 10:54 /home/hadoop/outputdir

hduser@frontend:/home/hadoop$ /usr/local/hadoop/bin/hadoop fs -cat /home/hadoop/outputdir

16/09/12 15:08:23 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

cat: `/home/hadoop/outputdir': Is a directory

hduser@frontend:/home/hadoop$ /usr/local/hadoop/bin/hadoop fs -cat /home/hadoop/outputdir1

16/09/12 15:08:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

cat: `/home/hadoop/outputdir1': No such file or directory

hduser@frontend:/home/hadoop$ export HADOOP_OPTS="$HADOOP_OPTS -Djava.library.path=/usr/local/hadoop/lib/native"

hduser@frontend:/home/hadoop$ /usr/local/hadoop/bin/hadoop fs -cat /home/hadoop/outputdir1

cat: `/home/hadoop/outputdir1': No such file or directory

hduser@frontend:/home/hadoop$ /usr/local/hadoop/bin/hadoop fs -cat outputdir/part-00000

1981 34

1984 38

1985 39

hduser@frontend:/home/hadoop$ /usr/local/hadoop/bin/hadoop fs -cat outputdir/part-00000/bin/hadoop dfs get output_dir /home/hadoop

```
cat: `outputdir/part-00000/bin/hadoop': No such file or directory

cat: `dfs': No such file or directory

cat: `get': No such file or directory

cat: `output_dir': Is a directory

cat: `/home/hadoop': Is a directory

hduser@frontend:/home/hadoop$  /usr/local/hadoop/bin/hadoop fs  -cat  outputdir/part-00000

1981    34

1984    38

1985    39
```

RESULT: Thus, the Map Reduce Program for ProcessUnits.java is executed successfully.

VIRTUAL BOX INSTALLATION

AIM:

To install the Oracle Virtual Box and to create VM.

INTRODUCTION:

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

VirtualBox is being actively developed with frequent releases and has an ever growing list of features, supported guest operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company: everyone is encouraged to contribute while Oracle ensures the product always meets professional quality criteria.

STEPS:

1. Check the distribution codename using the following command

   $ lsb_release -c

2. Edit the /etc/apt/sources.list file

```
deb http://download.virtualbox.org/virtualbox/debian trusty contrib
```

3. Download the oracle public key

```
$ wget http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc $ sudo apt-key add
oracle_vbox.asc
```

4. Install oracle virtual box

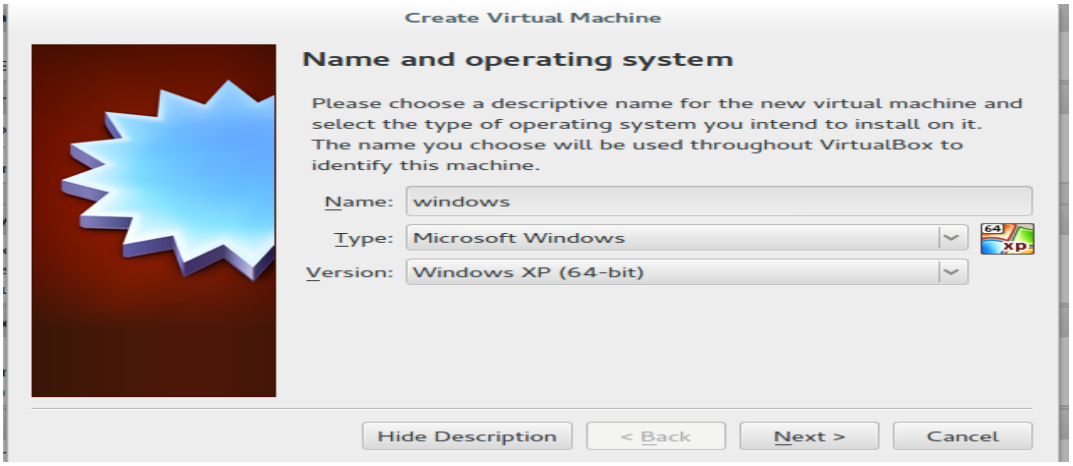
   $ sudo apt-get update

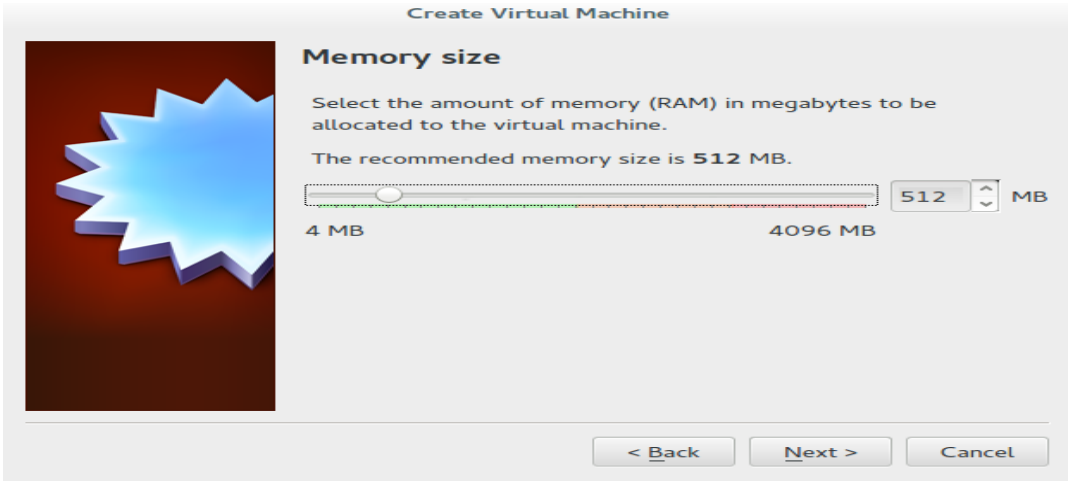
   $ sudo apt-get install virttualbox-5.0

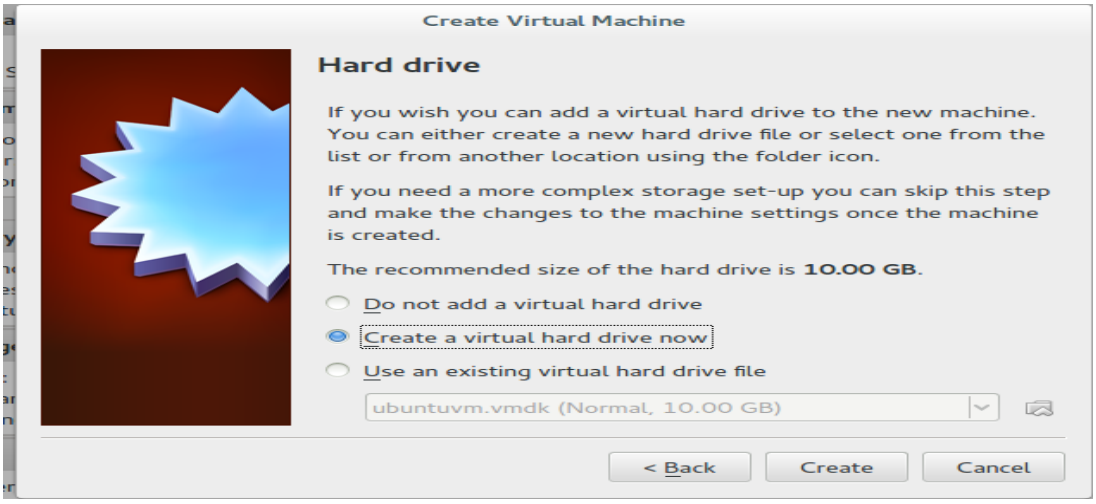
5. Start the virtualbox

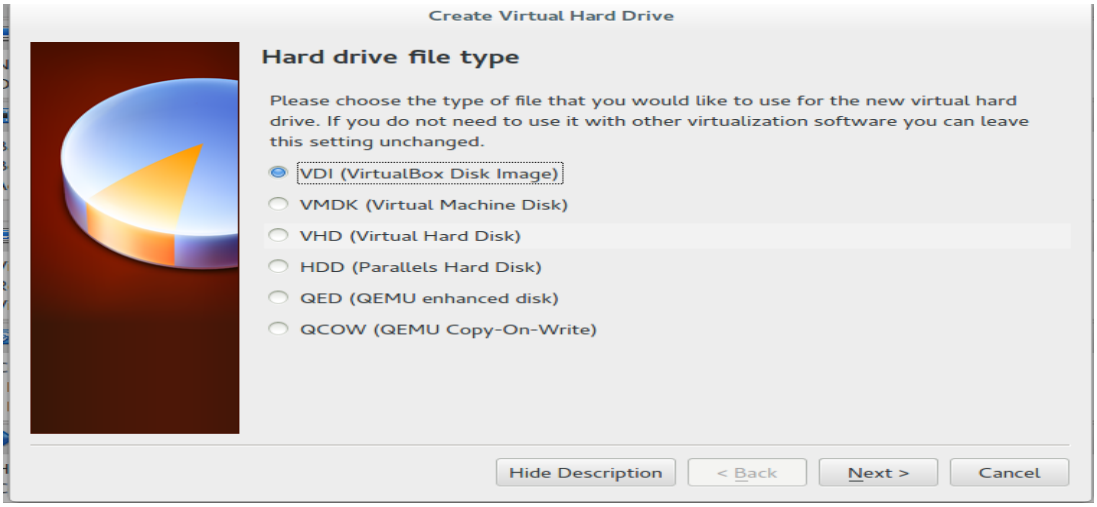
```
$ virtualbox
```

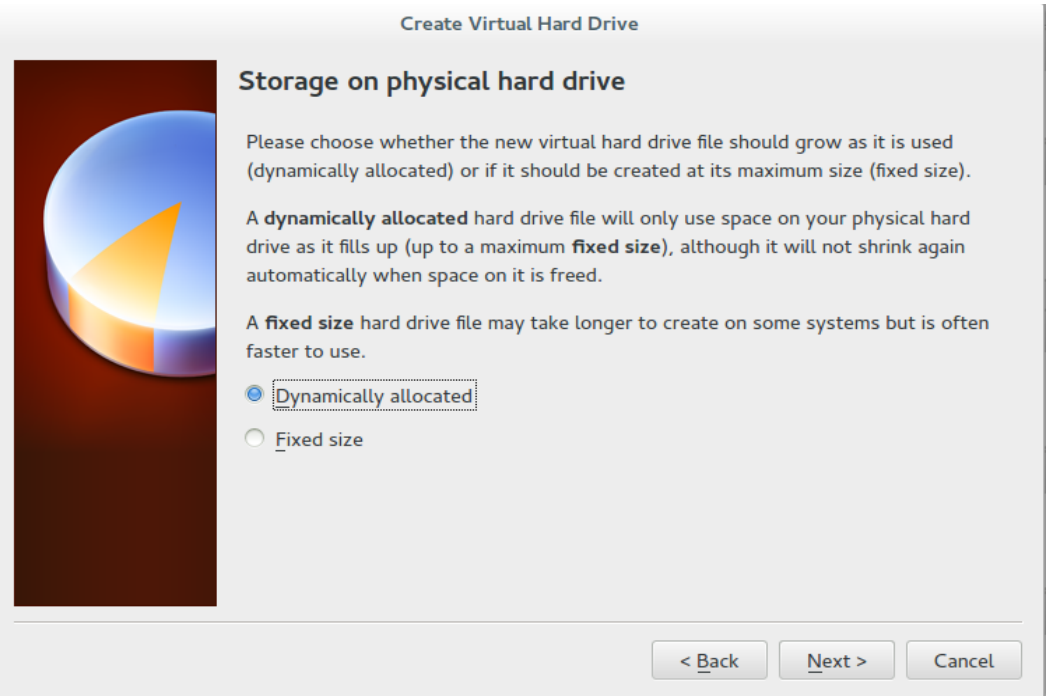
OUTPUT SCREENSHOTS:

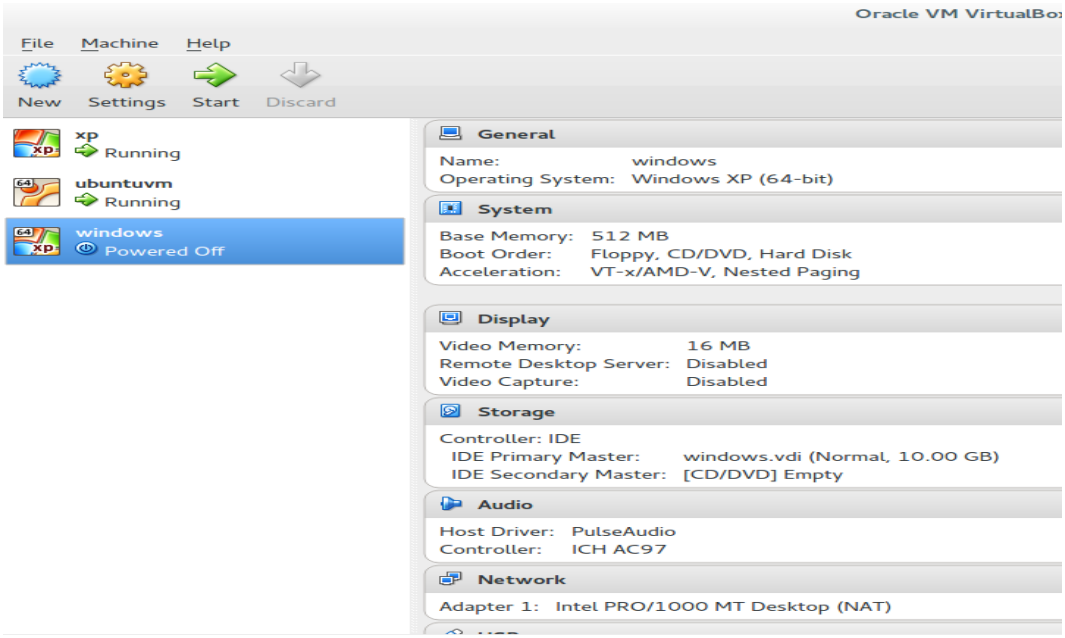




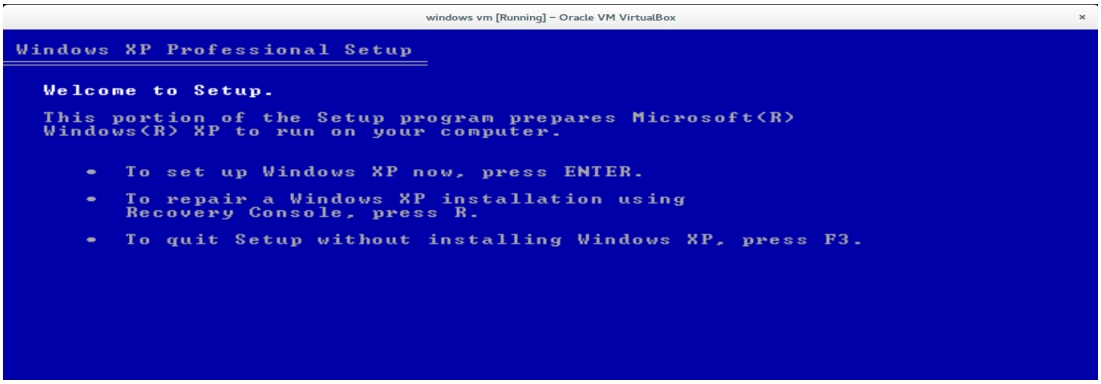

Create Virtual Machine
Hard drive
If you wish you can add a virtual hard drive to the new machine. You can either create a new hard drive file or select one from the list or from another location using the folder icon.
If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.
The recommended size of the hard drive is 10.00 GB.
Do not add a virtual hard drive
Create a virtual hard drive now
Use an existing virtual hard drive file
ubuntuvm.vmdk (Normal, 10.00 GB)
< Back
Create
Cancel

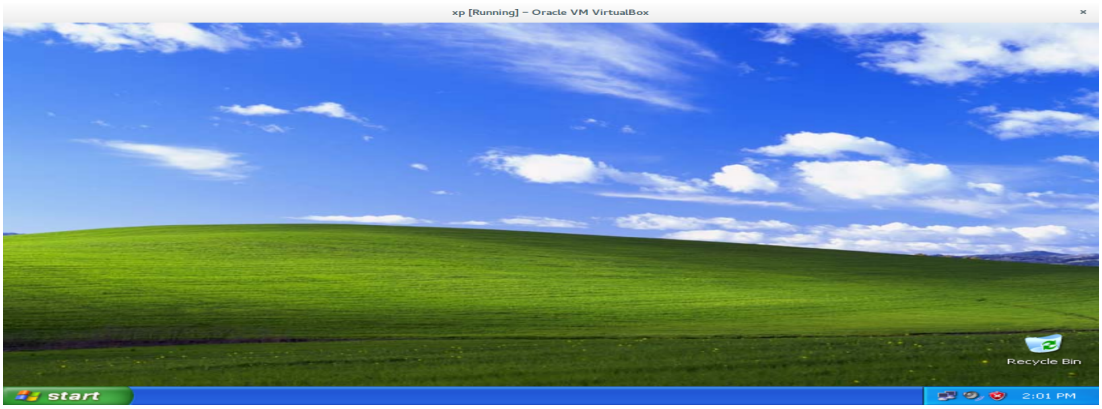

Create Virtual Hard Drive
Hard drive file type
Please choose the type of file that you would like to use for the new virtual hard drive. If you do not need to use it with other virtualization software you can leave this setting unchanged.
VDI (VirtualBox Disk Image)
VMDK (Virtual Machine Disk)
VHD (Virtual Hard Disk)
HDD (Parallels Hard Disk)
QED (QEMU enhanced disk)
QCOW (QEMU Copy-On-Write)
Hide Description
< Back
Next >
Cancel


Create Virtual Hard Drive
Storage on physical hard drive
Please choose whether the new virtual hard drive file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).
A dynamically allocated hard drive file will only use space on your physical hard drive as it fills up (up to a maximum fixed size), although it will not shrink again automatically when space on it is freed.
A fixed size hard drive file may take longer to create on some systems but is often faster to use.
Dynamically allocated
Fixed size
< Back
Next >
Cancel


Oracle VM VirtualBox
File
Machine
Help
New
Settings
Start
Discard
xp
Running
ubuntuvm
Running
windows
Powered Off
General
Name: windows
Operating System: Windows XP (64-bit)
System
Base Memory: 512 MB
Boot Order: Floppy, CD/DVD, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging
Display
Video Memory: 16 MB
Remote Desktop Server: Disabled
Video Capture: Disabled
Storage
Controller: IDE
IDE Primary Master: windows.vdi (Normal, 10.00 GB)
IDE Secondary Master: [CD/DVD] Empty
Audio
Host Driver: PulseAudio
Controller: ICH AC97
Network
Adapter 1: Intel PRO/1000 MT Desktop (NAT)

VIRTUAL MACHINE INSTALLATION





RESULT:

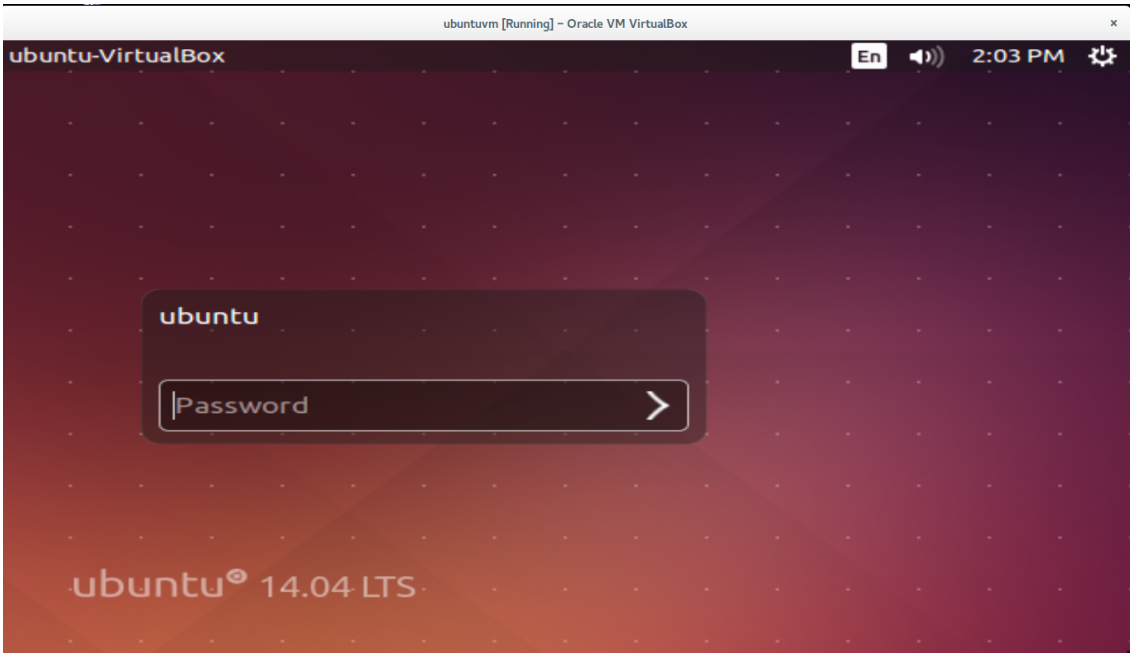
Hence, the Oracle Virtual Box is installed and VM is also created.

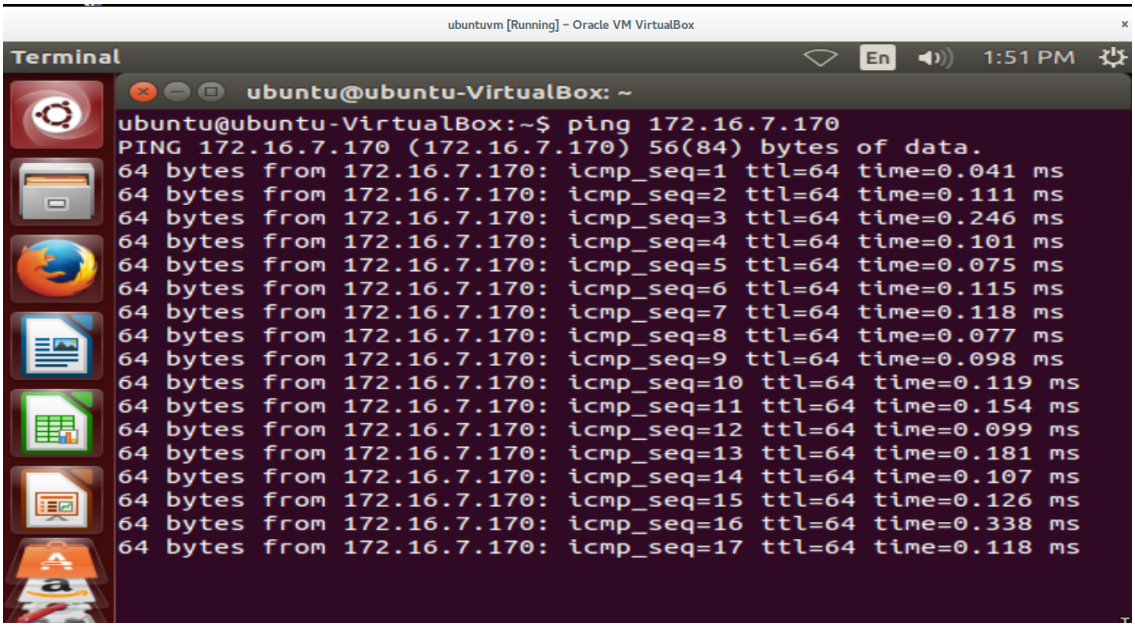
COMMUNICATION BETWEEN PHYSICAL MACHINE AND VIRTUAL MACHINE

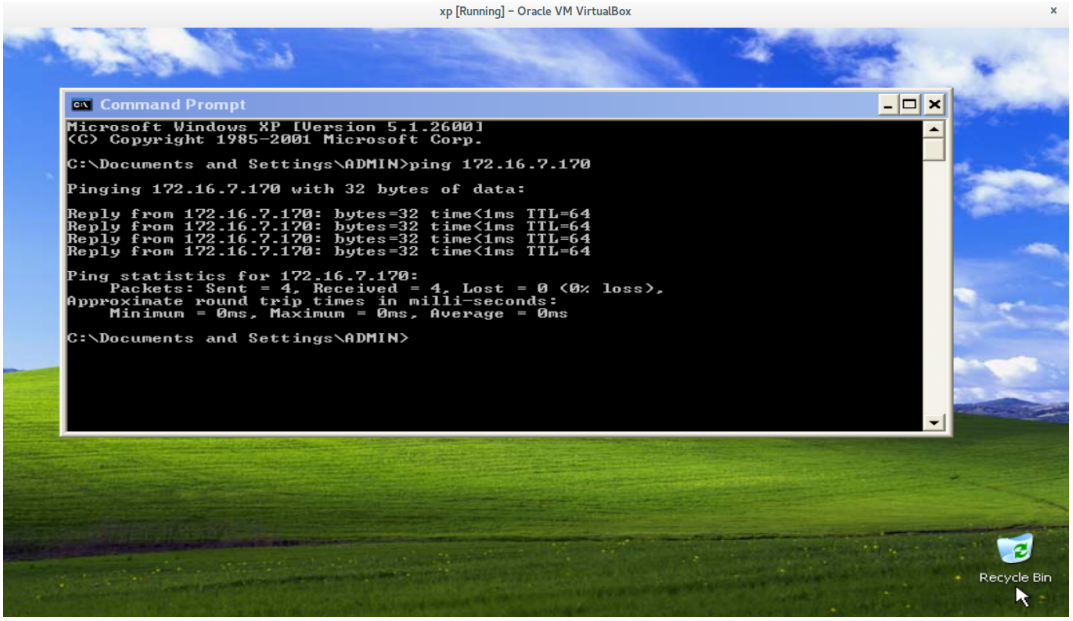
AIM:

To study and verify the communication between the physical machine and virtual machine.

SCREENSHOTS:







RESULT:

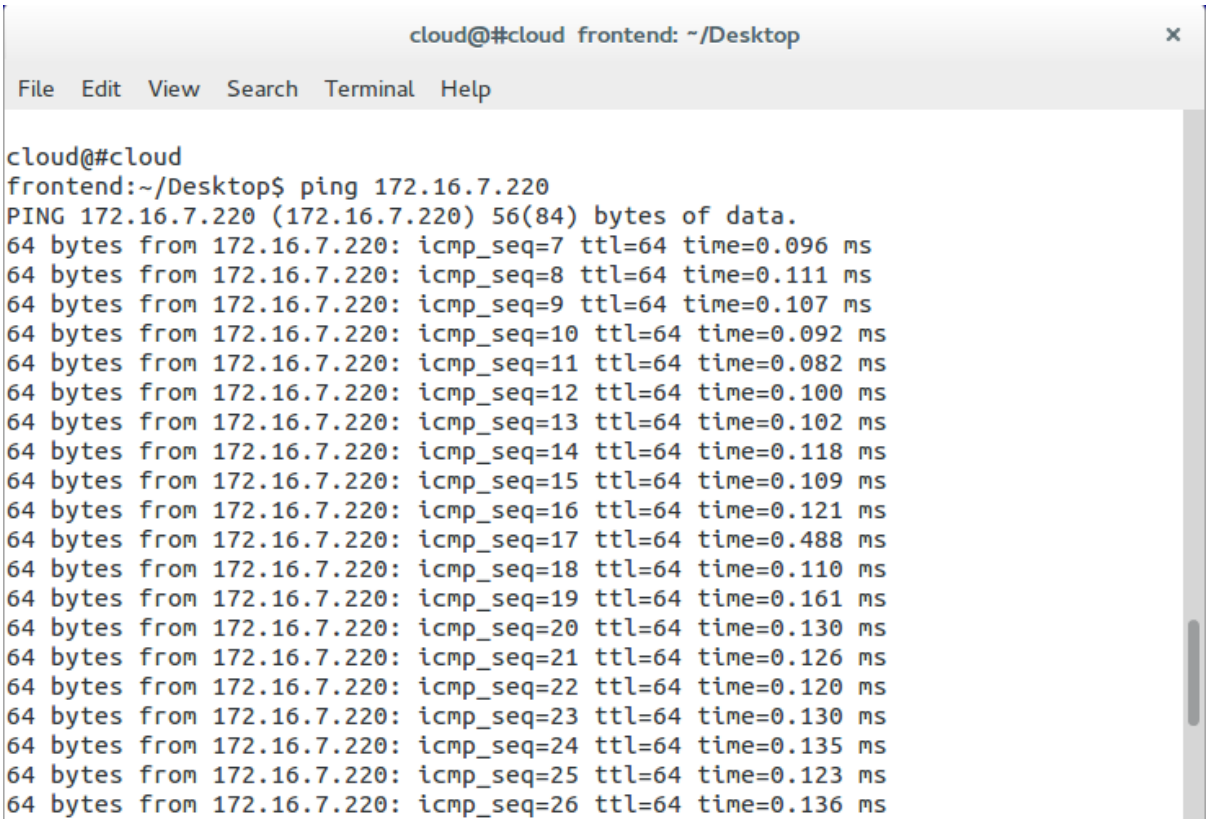
Thus, the communication between Physical Machine and Virtual Machine is studied.

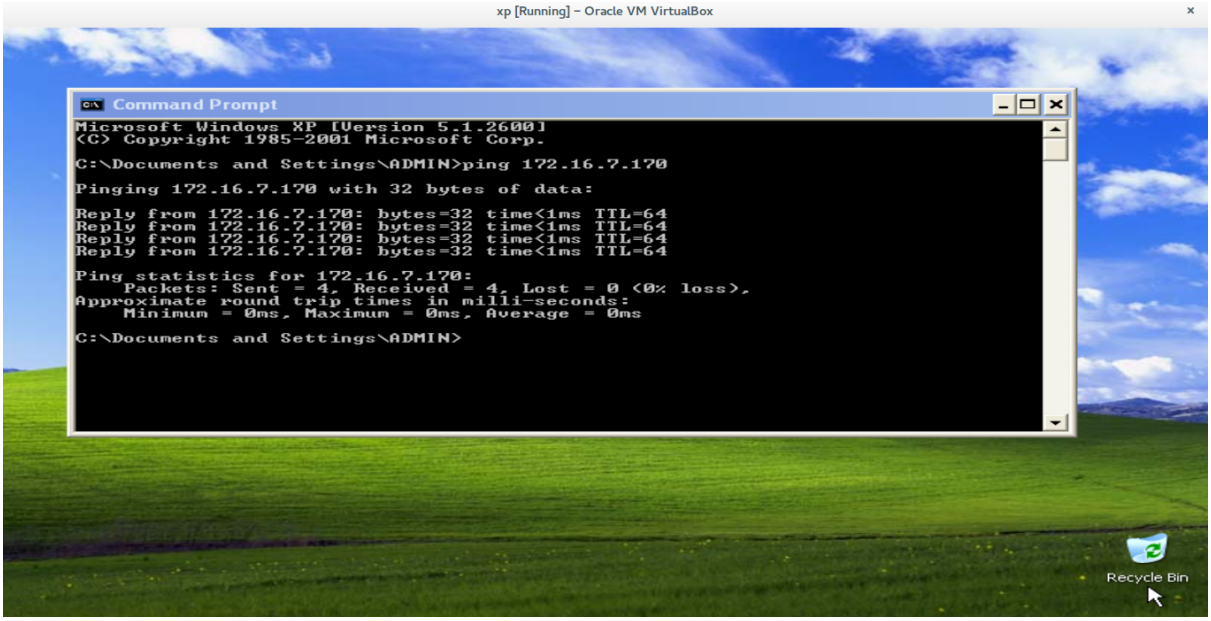
COMMUNICATION BETWEEN VIRTUAL MACHINE AND VIRTUAL MACHINE

AIM:

To study and verify the communication between the virtual machine and virtual machine.

SCREENSHOTS:





RESULT:

Thus, the communication between virtual machine and virtual machine is studied.

SIMPLE C PROGRAM IN VIRTUAL MACHINE

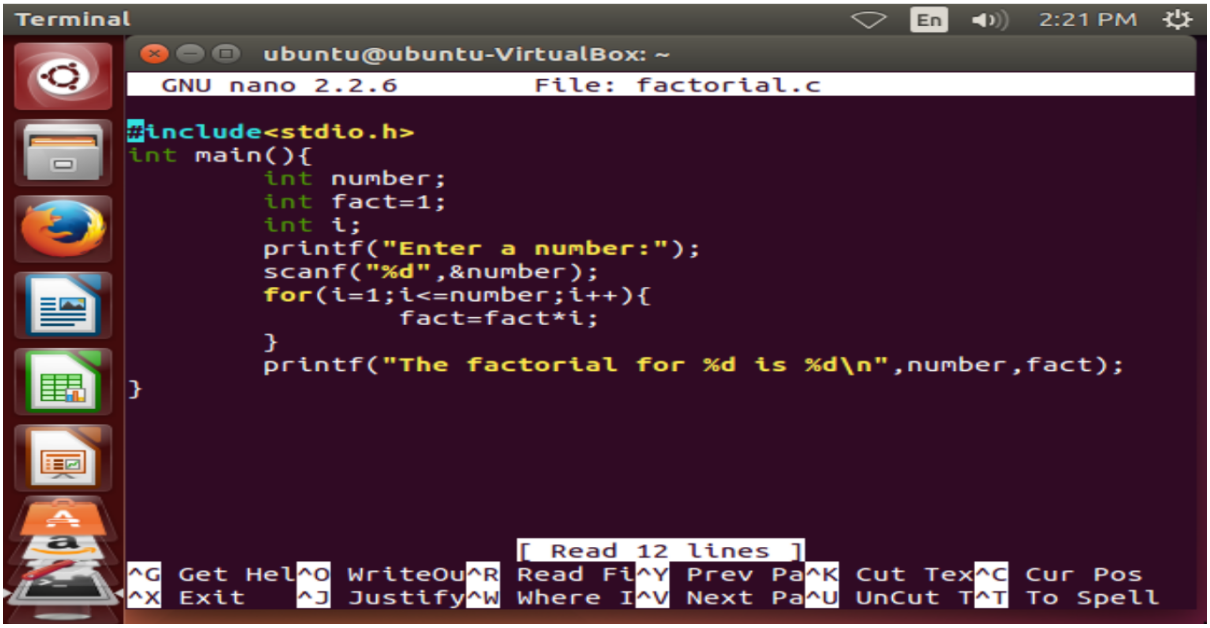
AIM:

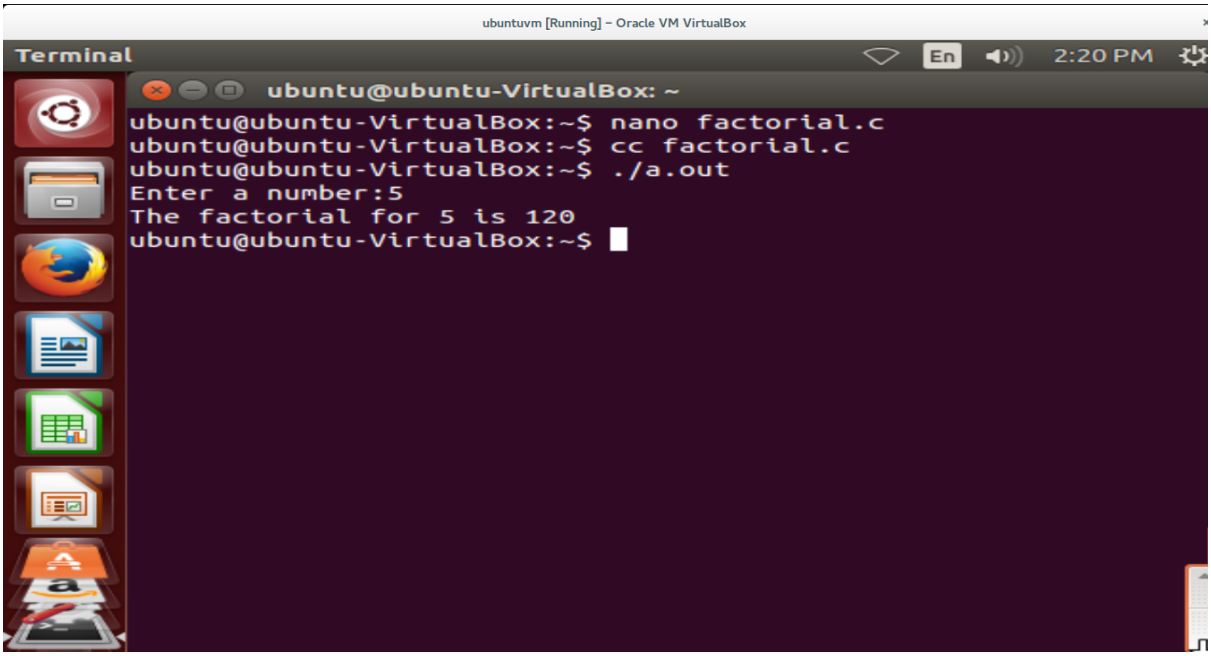
To write a simple C Program to find the factorial of a number in Virtual Machine.

PROGRAM:

```
#include<stdio.h>
int main()
{
int number;
int fact=1;
int i;
printf("Enter a number:");
scanf("%d",&number);
for(i=1;i<=number;i++)
{
Fact=fact*i;
}
printf("The factorial for %d is %d\n",number,fact);
return 0;
}
```

SCREENSHOTS:





RESULT:

Thus, a simple C Program to find the factorial of a number in Virtual Machine is executed.

SIMPLE JAVA PROGRAM IN VIRTUAL MACHINE

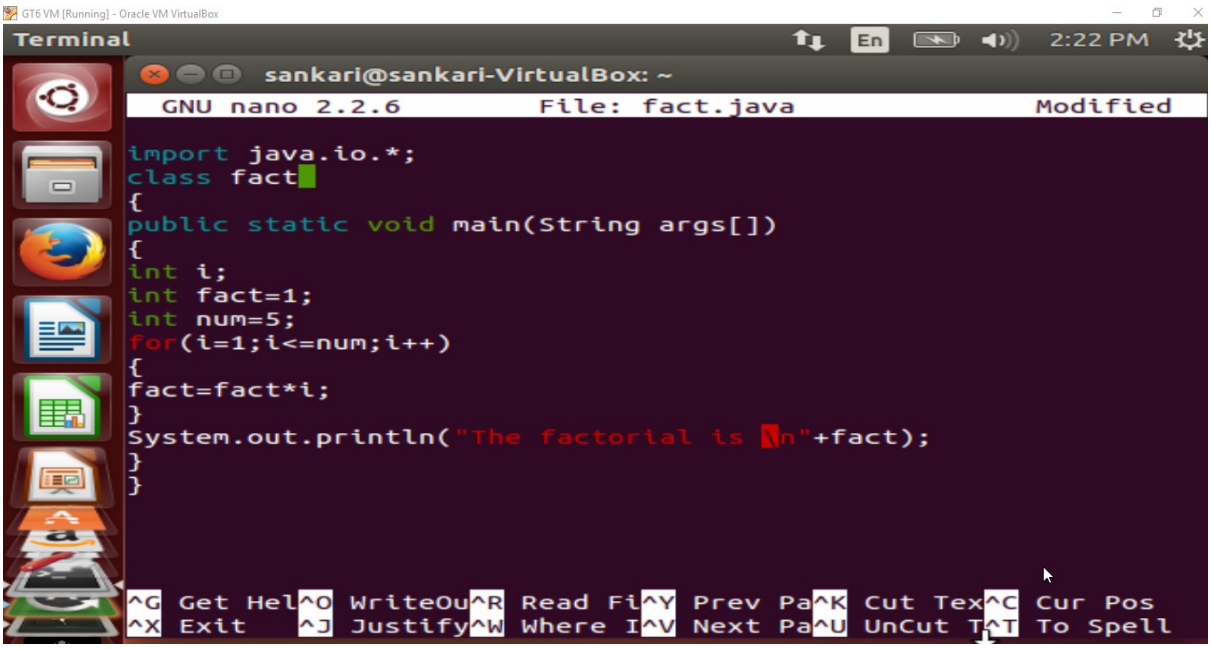
AIM:

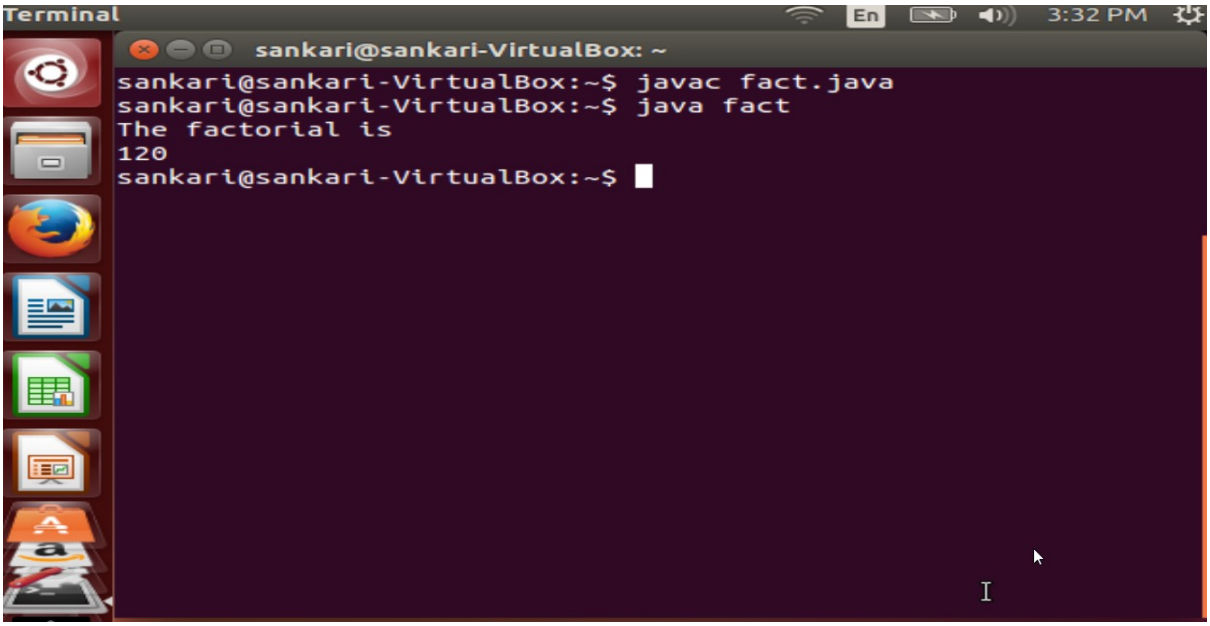
To write a simple JAVA Program to find the factorial of a number in Virtual Machine.

PROGRAM:

```
import java.io.*;
class fact
{
int num=5;
int fact=1;
int i;
for(i=1;i<=num;i++)
{
fact=fact*i;
}
System.out.println(“The factorial for is \n”,+fact);
}
}
```

SCREENSHOTS:





RESULT:

Thus, a simple JAVA Program to find the factorial of a number in Virtual Machine is executed