Assignment No. 11

Aim

KVM

Problem Definition

Perform a suitable assignment using Xen Hypervisor or equivalent open source to configure it. Give necessary GUI.

Learning Objectives

- To learn concept of virtualization
- To learn about KVM
- To learn how to install KVM

Learning Outcome

- Learnt the concept of virtualization
- Successfully installed of KVM

Software And Hardware Requirements

- Latest 64-BIT Version of Linux Operating System
- Modelio Software

Mathematical Model

Let S be the system of solution set for given problem statement such that,
S = { s, e, X, Y, F, DD, NDD, Su, Fu }
where,
s = start state
such that, y = { }
e = end state
such that, y = { KVM }
KVM = KVM Hypervisor installed

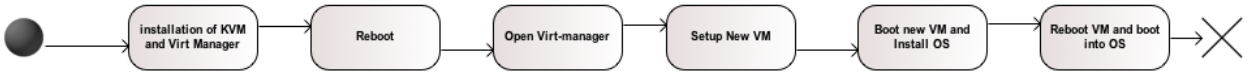
X = set of inputs
such that X = { x1, x2, x3, x4 }
where,
x1 = Allocated RAM
x2 = Video Memory
x3 = Virtual HDD size
x4 = CPU Threshold
Y = set of output
such that Y = { y1 }
where, y1 = Guest OS installed using KVM
F = set of function
such that F = { f1, f2, f3 }
where,
f1 = function to requirements from user
f2 = function to install Xen Hypervisor
f3 = function to start Xen
DD = Deterministic data
DD = { x1, x2, x3, x4 }
NDD = Nondeterministic data
NDD = { y1 }
Su = Success case

- KVM is installed successfully in machine
- Machine has minimum hardware specification to run Guest OS

Fu = Failure case

- KVM is not installed and setup properly
- Machine hardware specification is below minimum required one

State Diagram



Theory

Hypervisor

A hypervisor or virtual machine monitor (VMM) is a piece of computer software, firmware or hardware that creates and runs virtual machines.
There are two types of hypervisors:

- Type-1, native or bare-metal hypervisors:

  These hypervisors run directly on the host's hardware to control the hardware and to manage guest operating systems. For this reason, they are sometimes called bare metal hypervisors. A guest operating system runs as a process on the host.

- Type-2 or hosted hypervisors:

  These hypervisors run on a conventional operating system just as other computer programs do. Type-2 hypervisors abstract guest operating systems from the host operating system. VMware Workstation, VMware Player, VirtualBox and QEMU are examples of type-2 hypervisors.

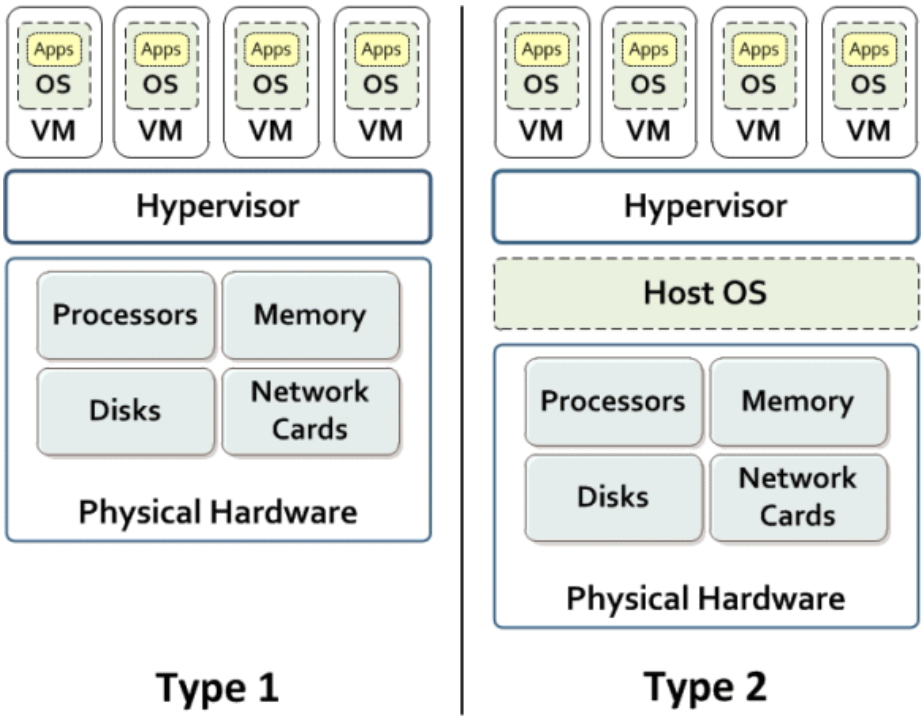


Fig : Type I and Type II Hypervisors

However, the distinction between these two types is not necessarily clear. Linux's Kernel - based Virtual Machine (KVM) and FreeBSD bhyve are kernel modules that effectively convert the host operating system to a type-1 hypervisor.

Kernel Based Virtual Machine (KVM):

Kernel-based Virtual Machine (KVM) is a virtualization infrastructure for the Linux kernel that turns it into a hypervisor. KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko.

Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, etc.

KVM is open source software. The kernel component of KVM is included in mainline Linux, as of 2.6.20. The userspace component of KVM is included in mainline QEMU, as of 1.3. By itself, KVM does not perform any emulation. Instead, it exposes the /dev/kvm interface, which a userspace host can then use to:

- Set up the guest VM's address space. The host must also supply a firmware image (usually a custom BIOS when emulating PCs) that the guest can use to bootstrap into its main OS
- Feed the guest simulated I/O
- Map the guest's video display back onto the host

Software Architecture

On Linux, QEMU versions 0.10.1 and later is one such userspace host. QEMU uses KVM when available to virtualize guests at near-native speeds, but otherwise falls back to software-only emulation.

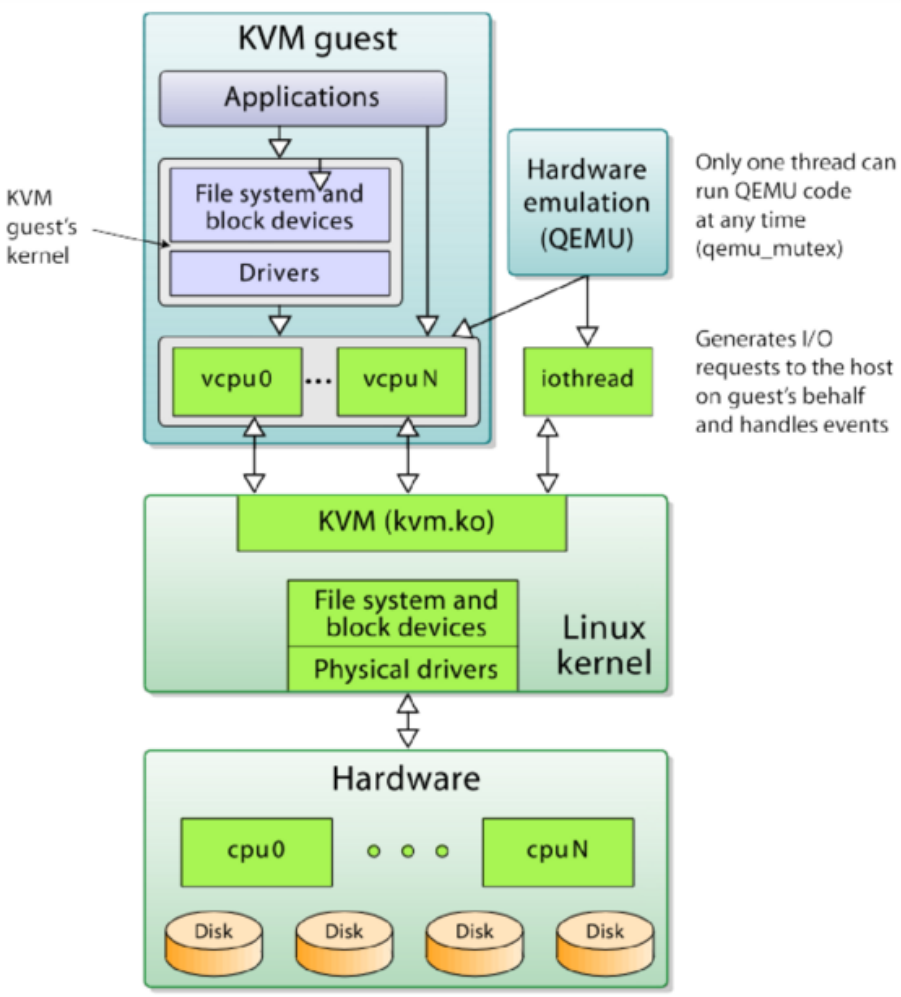


Fig : KVM Architecture

Output

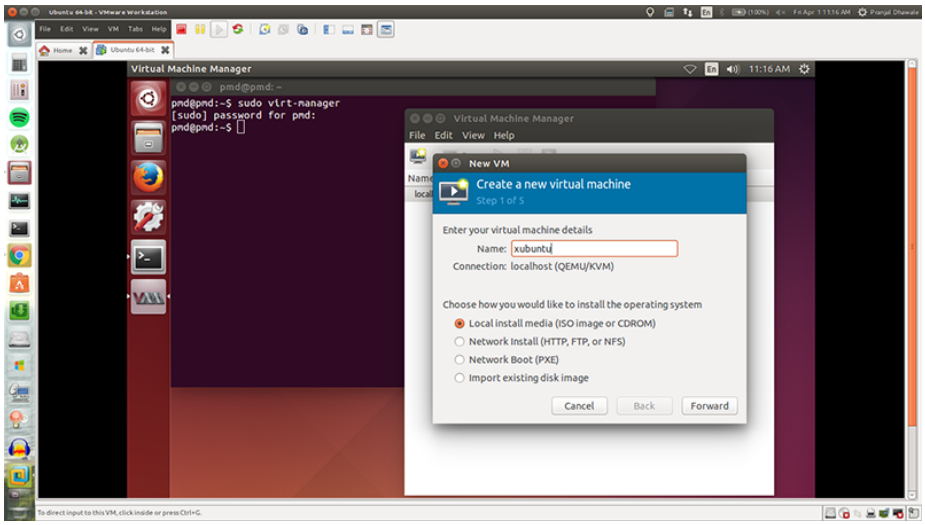


Fig : Create new VM

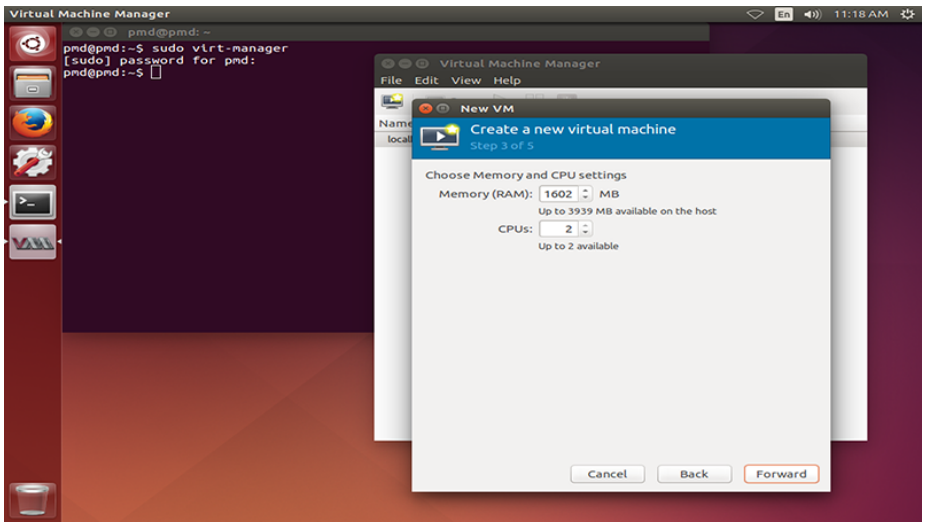


Fig : Specifications for VM

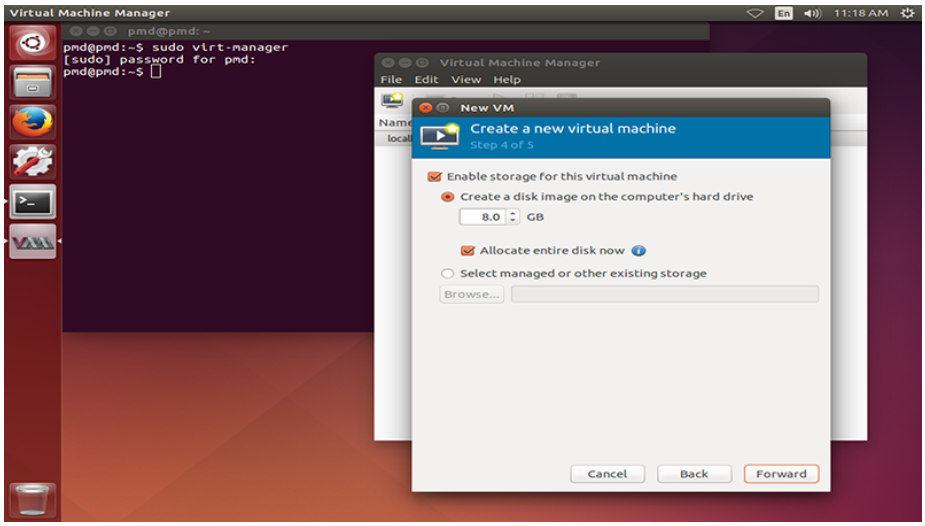


Fig : Specifications for VM

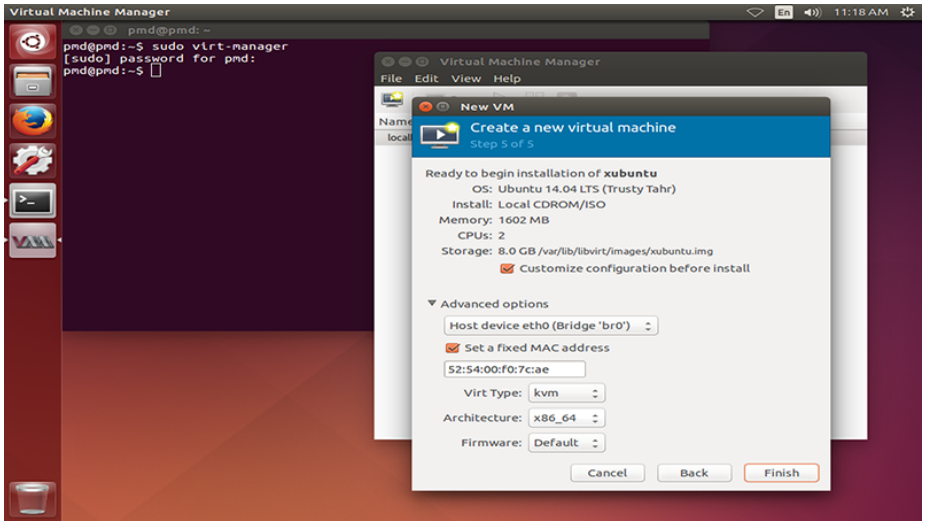


Fig : Specifications for VM

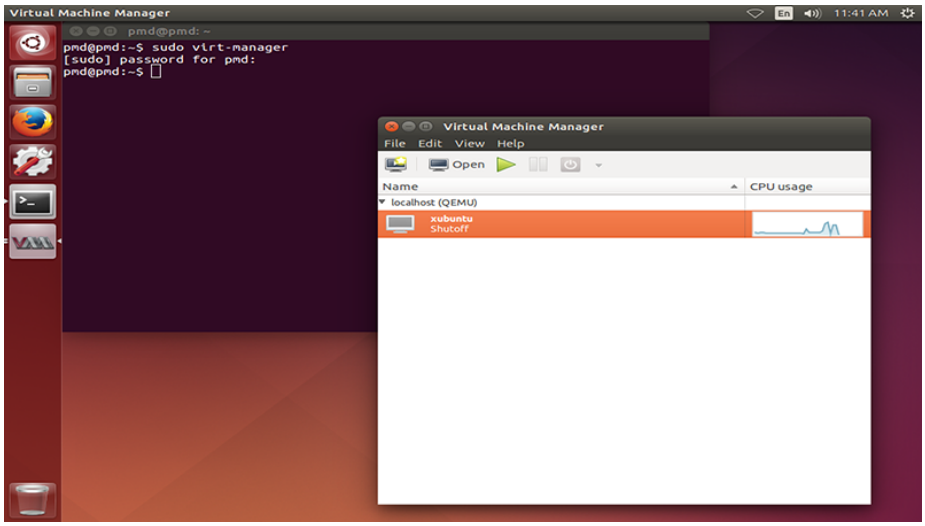
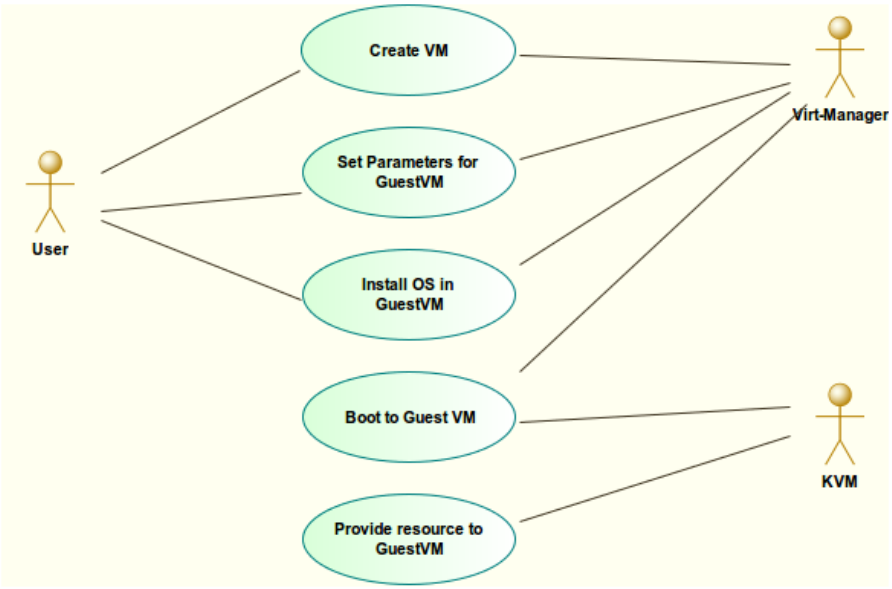


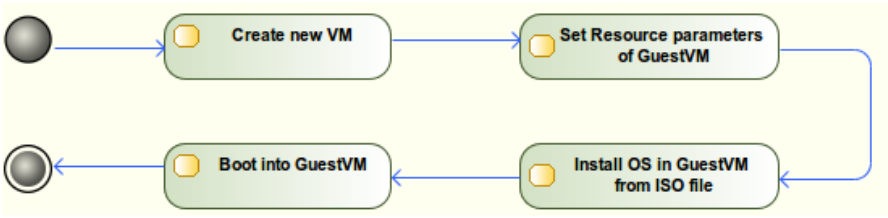
Fig : Newly Created VM

UML Diagrams

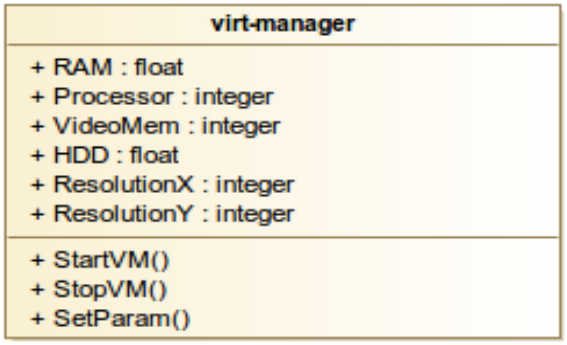
USE-CASE Diagram



Activity Diagram



Class Diagram



Conclusion

Thus, we understood the concepts of hypervisor and successfully implemented the KVM.